(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2017-199354
(P2017-199354A)
(43)公開日　平成29年11月2日(2017.11.2)

(51)Int.Cl.		F I		テーマコード（参考）
G06T 15/55	(2011.01)	G06T	15/55	5B080
G06T 17/20	(2006.01)	G06T	17/20	

審査請求 未請求　請求項の数 18　ＯＬ　外国語出願　（全 65 頁）

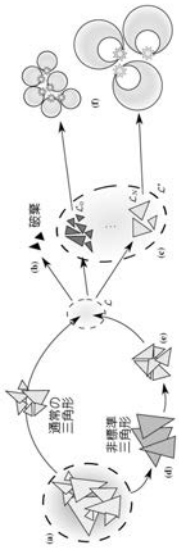
(21)出願番号　特願2017-32388(P2017-32388)
(22)出願日　平成29年2月23日(2017.2.23)
(31)優先権主張番号　16305221.0
(32)優先日　平成28年2月25日(2016.2.25)
(33)優先権主張国　欧州特許庁(EP)

(71)出願人　500102435
ダッソー　システムズ
ＤＡＳＳＡＵＬＴ　ＳＹＳＴＥＭＥＳ
フランス国　７８１４０　ベリジー　ビラクブレー　リュ　マルセル　ダッソー　１０
(74)代理人　110000752
特許業務法人朝日特許事務所
(72)発明者　ジル　ローラン
フランス国　９２３２０　シャティヨン　アベニュー　デ　ラ　ペ　７０
(72)発明者　シリル　デュラローンド
フランス国　９２３２０　シャティヨン　ル　ピエール　ブロソレット　８２　Ａ２号室

最終頁に続く

(54)【発明の名称】３Ｄシーンのグローバル・イルミネーションの描画

(57)【要約】　（修正有）
【課題】三次元シーンのグローバル・イルミネーションと描画するための、改良された方法を提供する。
【解決手段】三角形の集合と、１つまたは複数の直接光源とを含む３Ｄシーンを提供することと、前記集合の各三角形が、閾値以下の面積を有すると判定することと、確率法則を用いて、前記集合の各三角形に影響半径を割り当てることと、前記三角形を、それぞれの影響半径に応じて選別することによって、三角形の部分集合を得ることと、前記三次元シーンを、その三角形の集合へのライティングによって描画することを含む。描画には、前記三角形の部分集合の三角形が、その影響半径に応じて間接光源として用いられる。
【選択図】図３



【特許請求の範囲】
【請求項１】
コンピュータによって実施される、三次元シーンのグローバル・イルミネーションを描画するための方法であって、
三角形の集合Ｌと、１つまたは複数の直接光源とを含む三次元シーンを提供するステップ（Ｓ１０）と、
前記集合の各三角形t_iが、閾値S_o以下の面積を有すると判定するステップ（Ｓ３０）と、
確率法則を用いて、前記集合の各三角形に影響半径を割り当てるステップ（Ｓ５０）と、
前記三角形を、それぞれの影響半径に応じて選別すること（Ｓ５１０）によって、三角形の部分集合L^*を得るステップと、
前記三次元シーンを、その三角形の集合へのライティングによって描画するステップであって、前記三角形の部分集合L^*の三角形がその影響半径に応じて間接光源として用いられるような、描画するステップ（Ｓ８０）と
を含む方法。
【請求項２】
請求項１に記載の、コンピュータによって実施される方法であって、
前記集合の各三角形が前記閾値以下の面積を有すると判定するステップは、
三角形の集合Ｌを
それぞれが前記閾値以下の面積を有する三角形の第１の集合（Ｓ３１０）と、
それぞれが前記閾値よりも大きい面積を有する三角形の第２の集合（Ｓ３２０）と
に分割するステップと、
第２の集合の各三角形をテッセレーションするステップ（Ｓ３２２）であって、テッセレーションの結果として前記閾値以下の面積を有する三角形の第３の集合を得るステップを含み、
ここで、
確率法則を用いて前記集合の各三角形に影響半径を割り当てるステップは、確率法則を用いて前記第１および第３の集合の各三角形に影響半径を割り当てるステップを含み、
三角形の部分集合を取得するステップは、前記第１および第３の集合の三角形を、それらの影響半径に応じて選別することによって三角形の部分集合を取得するステップを含む
ことを特徴とする方法。
【請求項３】
請求項１～２のうちの１つに記載の、コンピュータによって実施される方法であって、
前記集合の各三角形が閾値以下の面積を有すると判定した後、
前記確率法則に従って、前記三角形の集合を、三角形のＮ＋２個の部分集合にランダムに分割するステップ（Ｓ５０）と、
三角形の前記Ｎ＋２個の部分集合のうち、前記三次元シーンの描画に対する寄与度が低いと判定された三角形を含む１つを破棄するステップ（Ｓ５１０）であって、残りのＮ＋１個の部分集合L^*が、破棄されていない三角形を含むステップと
をさらに含む
ことを特徴とする方法。
【請求項４】
請求項３に記載の、コンピュータによって実施される方法であって、
三角形の前記Ｎ＋２個の部分集合のうちの残りの部分集合L^*は、Ａ（t_i）を三角形t_iの表面積、S_oを閾値としたとき、

【数１】

$$\forall t_i \in \mathcal{L},\ P(t_i \in \mathcal{L}^*) = \frac{\mathcal{A}(t_i)}{\mathcal{S}_0}$$

となるような前記集合の三角形ｔ$_i$を含む
ことを特徴とする方法。
【請求項５】
請求項４に記載の、コンピュータによって実施される方法であって、
前記残りの部分集合Ｌ*の各三角形は、三角形が、
【数２】

$$P(t_i \in \mathcal{L}^k) = \begin{cases} \dfrac{\mathcal{A}(t_i)}{\mathcal{S}_N}, & \text{if } k = N \\ \dfrac{\mathcal{A}(t_i)}{\mathcal{S}_k} - \dfrac{\mathcal{A}(t_i)}{\mathcal{S}_{k+1}}, & \text{if } 0 \leqslant k < N \end{cases}$$

で定義される部分集合Ｌ^{k}内に存在する確率に従って、残りの部分集合Ｌ^{k}のうちの１つに送出され（Ｓ５２０）、
Ｓ$_k$の値はｋが増加すると増加する
ことを特徴とする方法。
【請求項６】
請求項５に記載の、コンピュータによって実施される方法であって、
Ｓ$_k$はＳ$_k$＝Ｓ$_o$μ^{k}を満たす集合であり、ここでμ＞１であり、μの値は［１；５］間の実数である
ことを特徴とする方法。
【請求項７】
請求項５に記載の、コンピュータによって実施される方法であって、
Ｓ$_k$はＳ$_k$＝Ｓ$_o$μ^{k}を満たす集合であり、ここでμの値は、三次元シーンの半径であるＲ$_{scene}$を用いて、
【数３】

$$\mu > \sqrt[N]{4\frac{\mathcal{R}_{scene}^2}{S_o}}$$

で定義される
ことを特徴とする方法。
【請求項８】
請求項５～７のうちの１つに記載の、コンピュータによって実施される方法であって、
前記残りの部分集合Ｌ*の各三角形が送出された後、
前記残りの部分集合Ｌ*の各三角形について単一の仮想点光源を算出するステップ（Ｓ６０）を含み、
ここで、確率法則を用いて前記集合の各三角形に影響半径を割り当てるステップはさらに、
前記残りの部分集合Ｌ*の各三角形の仮想点光源の出射放射輝度を算出するステップ（Ｓ７０）を含み、前記算出は、三角形が送出された前記部分集合に従って実行される
ことを特徴とする方法。
【請求項９】
請求項８に記載の、コンピュータによって実施される方法であって、

各三角形の仮想点光源について出射放射輝度を算出するステップは、
仮想点光源の寄与度が大きい三次元シーン内の点の集合を表す入れ子状のボール群$B_h(t_i)$を算出するステップと、
前記Ｎ＋２個の部分集合のうち、残りの部分集合のそれぞれについて、残りの部分集合上のサポート関数の集合$(f^k)_k$が、入れ子状のボール群$B_h(t_i)$の境界に亘って一定である１の分割を形成するようなサポート関数$f^k(t_i, x)$を算出するステップと
を含む
ことを特徴とする方法。
【請求項１０】
請求項９に記載の、コンピュータによって実施される方法であって、前記サポート関数の集合$(f^k)_k$の基底関数はＢスプラインである
ことを特徴とする方法。
【請求項１１】
請求項９または１０に記載の、コンピュータによって実施される方法であって、$B_h(t_i)$は、
【数４】

$$\mathcal{B}_h(t_i) = \{x \epsilon \mathbb{R}^3 \; s.t. \frac{||x-y_i||}{\langle \vec{n_x}, \; \bar{x}y_i \rangle^+} \leq D(h) = \frac{1}{\pi}\sqrt{\frac{\rho_x \rho_i E(t_i)}{h}}\}$$

で定義され、
ここで、ｘは法線
【数５】

$$\vec{n_x}$$

を有する三次元シーンの点であり、
y_iは三角形t_i上の仮想点光源の中心となる点であり、
ρ_xは点ｘにおけるアルベドであり、
ρ_iは点y_iにおけるアルベドであり、
$E(t_i)$は、三角形t_iへの直接放射照度である
ことを特徴とする方法。
【請求項１２】
請求項９～１１のうちの１つに記載の、コンピュータによって実施される方法であって、
前記描画ステップにおいて、入れ子状のボール群$B_h(t_i)$の球面を描画するときに、サポート関数の集合$(f^k)_k$が、スプラッティング演算を実行するのに用いられる
ことを特徴とする方法。
【請求項１３】
請求項１～１２のうちの１つに記載の、コンピュータによって実施される方法であって、
前記閾値S_oは、
【数６】

$$S_o = 4\pi \frac{D_{near}^2}{N_{Avg}}$$

で定義され、
ここで、N_{Avg}は、仮想点光源の平均個数であり、

D_{near}は、前記３Ｄシーンの画素とＶＰＬとの間の最短距離である
ことを特徴とする方法。
【請求項１４】
請求項１～１３のうちの１つに記載の、コンピュータによって実施される方法であって、
前記集合の各三角形t_iが閾値S_o以下の面積を有すると判定する前に、
前記集合の三角形t_iの各頂点について単一のランダムな整数を生成する（Ｓ２０）ステップをさらに含む
ことを特徴とする方法。
【請求項１５】
請求項２と組み合わせた請求項１４に記載の、コンピュータによって実施される方法であって、
三角形の各頂点について単一のランダムな整数を算出するステップは、
前記第３の集合の三角形の各副頂点について、当該副頂点の重心座標を決定するステップと、
前記重心座標を用いてノイズ・テクスチャ内のランダムな値を取得するステップ（Ｓ２１０）と、
前記取得したランダムな値を用いて前記第３の集合の三角形の前記各副頂点の前記単一のランダムな整数を生成するステップ（Ｓ２２０）と
を含む
ことを特徴とする方法。
【請求項１６】
請求項１～１５のいずれかに記載の方法を実行するための命令を含む、グラフィック・ライブラリ・コンピュータプログラム。
【請求項１７】
請求項１６に記載のコンピュータプログラムを記録したメモリに接続されたプロセッサと、グラフィック処理ユニットを有するグラフィック・カードとを備えるシステム。
【請求項１８】
請求項１７に記載のシステムであって、前記グラフィック・カードは、
前記集合の各三角形t_iが閾値S_o以下の面積を有すると判定するジオメトリ・シェーダ・ユニットと、
前記集合の三角形t_iをテッセレーションするためのテッセレーション・ユニットとを備え、
前記メモリ上に記録された前記コンピュータプログラムは、請求項１４または１５に記載の前記単一のランダムな整数の生成を実行するために前記テッセレーション・ユニットを構成するように適合されている
ことを特徴とするシステム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明はコンピュータプログラムおよびシステムの分野に関連し、より詳細には、通常ＣＡＤアプリケーションのシナリオにおいて見られる複雑なシーンのような大きなモデルの文脈における、三次元シーンのグローバル・イルミネーションを描画するための方法、システム、およびプログラムに関連する。
【背景技術】
【０００２】
オブジェクトの設計、エンジニアリング、製造のため、多数のシステムおよびプログラムが市場に提供されている。ＣＡＤは、コンピュータ支援設計（Ｃｏｍｐｕｔｅｒ－Ａｉｄｅｄ　Ｄｅｓｉｇｎ）の略語であり、例えば、オブジェクトを設計するためのソフトウェア・ソリューションに関する。ＣＡＥは、コンピュータ支援エンジニアリング（Ｃｏｍ

ｐｕｔｅｒ－Ａｉｄｅｄ　Ｅｎｇｉｎｅｅｒｉｎｇ）の略語であり、例えば、将来の製品の物理的挙動をシミュレーションするためのソフトウェア・ソリューションに関する。ＣＡＭは、コンピュータ支援製造（Ｃｏｍｐｕｔｅｒ－Ａｉｄｅｄ　Ｍａｎｕｆａｃｔｕｒｉｎｇ）の略語であり、例えば、製造工程および動作を定義するためのソフトウェア・ソリューションに関する。このようなコンピュータ支援設計システムにおいて、グラフィカル・ユーザ・インターフェースは、技術の効率に関して、重要な役割を果たす。これらの技術は、製品ライフサイクル管理（Ｐｒｏｄｕｃｔ　Ｌｉｆｅｃｙｃｌｅ　Ｍａｎａｇｅｍｅｎｔ：　ＰＬＭ）システムに組み込むことができる。ＰＬＭとは、企業が、拡張エンタープライズの概念全体にわたって、製品データを共有し、共通の工程を適用し、構想に始まり製品寿命の終わりに至る製品開発のための企業知識を活用するのを支援するビジネス戦略を指す。ダッソー・システムズが提供するＰＬＭソリューション（製品名ＣＡＴＩＡ、ＥＮＯＶＩＡ、ＤＥＬＭＩＡ）は、製品エンジニアリング知識をオーガナイズするエンジニアリング・ハブ、製品エンジニアリング知識を管理する製造ハブ、およびエンジニアリング・ハブと製造ハブの両方に対するエンタープライズ統合と接続を可能にするエンタープライズ・ハブを提供する。全てのシステムは、製品、工程、リソースを結ぶオープンなオブジェクトモデルを提供し、最適化された製品定義、製造準備、生産およびサービスを推進する、動的な知識ベースの製品作成および意思決定支援を可能にする。

【０００３】

光輸送のシミュレーションは、現実的な画像合成の重要な要素である。この現象に関連する研究作業の根幹は、「グローバル・イルミネーション」と呼ばれ、物理法則がよく知られているにもかかわらず演算コストが高く、さらにはリアルタイムシナリオにとって重要な結果をもたらすため、依然として困難な課題である。ＫＡＪＩＹＡ，Ｊ．Ｔ．１９８６，Ｔｈｅ　ｒｅｎｄｅｒｉｎｇ　ｅｑｕａｔｉｏｎ，ＡＣＭ　Ｓｉｇｇｒａｐｈ　Ｃｏｍｐｕｔｅｒ　Ｇｒａｐｈｉｃｓ，ｖｏｌ．２０，ＡＣＭ，１４３－１５０で論じられている描画方程式の再帰的性質の陰に隠れているが、グローバル・イルミネーションのシミュレーションは、特殊効果、プリビジュアライゼーション、アニメーション映画制作、およびビデオゲームに高度な現実感をもたらすために、多くのアプローチで取り組まれてきた。

【０００４】

現在、２つの研究が区別できる。１つ目は、視覚的に説得力のあるグローバル・イルミネーションの近似を迅速に提供することを目指す「インタラクティブ・レンダリング」である。２つ目は、可能な限り物理学に近いソリューションを目標とする「オフライン・レンダリング」である。

【０００５】

「インタラクティブ・レンダリング」のリアルタイムパフォーマンスを達成するために使用できるさまざまな戦略の中で、遠隔入射放射輝度の低周波挙動を利用することが、効果的であると証明されている。例えば、ＷＡＬＴＥＲ，Ｂ．，ＦＥＲＮＡＮＤＥＺ，Ｓ．，ＡＲＢＲＥＥ，Ａ．，ＢＡＬＡ，Ｋ．，ＤＯＮＩＫＩＡＮ，Ｍ．，およびＧＲＥＥＮＢＥＲＧ，Ｄ．Ｐ．２００５．Ｌｉｇｈｔｃｕｔｓ：ａ　ｓｃａｌａｂｌｅ　ａｐｐｒｏａｃｈ　ｔｏ　ｉｌｌｕｍｉｎａｔｉｏｎ．ＡＣＭ　Ｔｒａｎｓａｃｔｉｏｎｓ　ｏｎ　Ｇｒａｐｈｉｃｓ（ＴＯＧ），ｖｏｌ．２４，ＡＣＭ，１０９８－１１０７で論じられている階層的放射輝度キャッシュ法は、マルチスケールの放射輝度関数を格納するシーンのジオメトリ上のツリーを算出することによってこの特性を利用する。間接照明が、シーンのサーフェス上に配置された仮想点光源（ＶＰＬ）によってモデル化されると仮定すると、このツリーのすべての内部ノードは、その関連するサブツリーの代表的な放射輝度応答を提供し、離れた場所から放射輝度が評価されるとすぐに、その代替物として使用される。ノードを集めることは、その子ノードを破棄することにつながるため、所与の点（例えば、投影されていない画像画素）における入射放射輝度を評価するために使用されるノードの集合は、ライト・カットと呼ばれる。

【０００６】

ライト・カット収集工程の高性能実装が多数提案されているが、それらの大部分はツリーを予め算出して維持することに依存しており、大規模なメモリと計算コストを要する可能性がある。

【０００７】

Ｍｉｔｔｒｉｎｇほか（ＭＩＴＴＲＩＮＧ，Ｍ．２００７．Ｆｉｎｄｉｎｇ ｎｅｘｔ ｇｅｎ：Ｃｒｙｅｎｇｉｎｅ ２．ＡＣＭ ＳＩＧＧＲＡＰＨ ２００７ ｃｏｕｒｓｅｓ，ＡＣＭ，９７－１２１）は、シーンの実際の（大きい可能性がある）ジオメトリに対して経済的かつランダムアクセスが可能な代替物として深度バッファを使用し、スクリーン空間内のライト・キャッシュをパラメータ化する、アンビエント・オクルージョン近似法を導入した。その他の様々な方法では、いくつかの照明効果の計算上の複雑さを低減するために、スクリーン空間近似を利用する。しかしながら、そのようなアプローチは、リアルタイムで動的なパフォーマンスが実現できるものの、シーンの完全なジオメトリ、すなわち最初の深度レイヤを超えた、視錐台の外のジオメトリを明らかにするのに、深度剥離および複数ビューの描画に依拠するため、すぐにネイティブスピードに悪影響を及ぼす。

【０００８】

スクリーン空間のアプローチとは対照的に、オブジェクト空間での間接イルミネーションの解決策では、ＧＰＵ負荷の少ないライト・キャッシュを犠牲にして、そのようなビューに依存したアーチファクトを回避する。例えば、ＫＥＬＬＥＲ，Ａ．１９９７．Ｉｎｓｔａｎｔ ｒａｄｉｏｓｉｔｙ．Ｉｎ Ｐｒｏｃｅｅｄｉｎｇｓ ｏｆ ｔｈｅ ２４ｔｈ ａｎｎｕａｌ ｃｏｎｆｅｒｅｎｃｅ ｏｎ Ｃｏｍｐｕｔｅｒ ｇｒａｐｈｉｃｓ ａｎｄ ｉｎｔｅｒａｃｔｉｖｅ ｔｅｃｈｎｉｑｕｅｓ，ＡＣＭ Ｐｒｅｓｓ／Ａｄｄｉｓｏｎ－Ｗｅｓｌｅｙ Ｐｕｂｌｉｓｈｉｎｇ Ｃｏ．，４９？５６で論じられているインスタント・ラジオシティー（Ｉｎｓｔａｎｔ Ｒａｄｉｏｓｉｔｙ：ＩＲ）法では、主光源によって照らされたジオメトリ上に直接生成された、仮想点光源（Ｖｉｒｔｕａｌ Ｐｏｉｎｔ Ｌｉｇｈｔｓ：ＶＰＬｓ）と呼ばれる、二次的点光源の集合が用いられる。このようにＶＰＬの集合はシーンの間接照明の離散表現として機能し、光の反射を近似する際の演算を大幅に減らすことができる。しかしながら、大規模なデータへと拡張するには、膨大な量のＶＰＬが必要となる。しかし、非常に多くのＶＰＬを生成しシェーディングを行うためのコストは、動的シーンにおいては法外に高いため、困難である。

【０００９】

最後に、ＮＡＬＢＡＣＨ，Ｏ．，ＲＩＴＳＣＨＥＬ，Ｔ．，ＡＮＤ ＳＥＩＤＥＬ，Ｈ．－Ｐ．２０１４．Ｄｅｅｐ ｓｃｒｅｅｎ ｓｐａｃｅ．Ｉｎ Ｐｒｏｃｅｅｄｉｎｇｓ ｏｆ ｔｈｅ １８ｔｈ ｍｅｅｔｉｎｇ ｏｆ ｔｈｅ ＡＣＭ ＳＩＧＧＲＡＰＨ Ｓｙｍｐｏｓｉｕｍ ｏｎ Ｉｎｔｅｒａｃｔｉｖｅ ３Ｄ Ｇｒａｐｈｉｃｓ ａｎｄ Ｇａｍｅｓ，ＡＣＭ，７９－８６．２０１４で論じられているディープ・スクリーン空間（Ｄｅｅｐ Ｓｃｒｅｅｎ Ｓｐａｃｅ（ＤＳＳ））では、スクリーン空間とオブジェクト空間の両方の放射輝度キャッシングの利点を利用することが提案されている。オブジェクト空間による戦略と同様、この方法では、画像に依然として影響を与えるかもしれない閉塞ジオメトリ上においても、オン・サーフェスＶＰＬを生成する。そしてスクリーン空間のアプローチと同様、ラスタライザの代わりにテッセレータ・ユニットがＶＰＬを生成するサーフェス・サンプラとして使用され、ネイティブＧＰＵサポートから利益を得る。ＤＳＳは小規模から中規模のシーンをうまく描画することができるが、より大きなものに対応することはできず、リアルタイムの制約により、ジオメトリを精緻化させるのではなく、間引くことが必要になる。

【００１０】

このような文脈において、三次元シーンのグローバル・イルミネーションを描画するための、改良された方法が依然として求められている。

【発明の概要】

【００１１】

したがって、コンピュータによって実施される、三次元シーンのグローバル・イルミネーションを描画（レンダリング）するための方法が提供される。本方法は、
・三角形の集合Ｌと、１つまたは複数の直接光源とを含む三次元シーンを提供するステップと、
・前記集合の各三角形t_iが閾値S_o以下の面積を有すると判定するステップと、
・確率法則を用いて、前記集合の各三角形に影響半径を割り当てるステップと、
・前記三角形を、それぞれの影響半径に応じて選別することによって、三角形の部分集合L^*を得るステップと、
・前記三次元シーンを、その三角形の集合へのライティングによって描画するステップであって、前記三角形の部分集合L^*の三角形がその影響半径に応じて間接光源として用いられるような、描画するステップとを含む。

【００１２】

本方法は、以下のうちの１つまたは複数を含んでいてもよい。
・前記集合の各三角形が前記閾値以下の面積を有すると判定するステップは、前記三角形の集合Ｌを、それぞれが前記閾値以下の面積を有する三角形の第１の集合と、それぞれが前記閾値よりも大きい面積を有する三角形の第２の集合とに分割するステップと、第２の集合の各三角形をテッセレーションして、テッセレーションの結果として前記閾値以下の面積を有する三角形の第３の集合を得るステップを含み、ここで、確率法則を用いて前記集合の各三角形に影響半径を割り当てるステップは、確率法則を用いて前記第１および第３の集合の各三角形に影響半径を割り当てるステップを含み、三角形の部分集合を取得するステップは、前記第１および第３の集合の三角形を、それらの影響半径に応じて選別することによって三角形の部分集合を取得するステップを含む。

【００１３】

・前記集合の各三角形が閾値以下の面積を有すると判定した後、前記三角形の集合を、前記確率法則に従って、前記三角形の集合を、三角形のＮ＋２個の部分集合にランダムに分割するステップと、三角形の前記Ｎ＋２個の部分集合のうち、前記３Ｄシーンの描画に対する寄与度が低いと判定された三角形を含む１つを破棄するステップであって、残りのＮ＋１個の部分集合は、破棄されていない三角形を含むような、破棄するステップとを含む。

【００１４】

・三角形の前記Ｎ＋２個の部分集合のうちの残りの部分集合L^*は、Ａ（t_i）を三角形t_iの表面積、S_oを閾値としたとき、

【数１】

$$\forall t_i \in \mathcal{L},\ \ P(t_i \in \mathcal{L}^*) = \frac{\mathcal{A}(t_i)}{\mathcal{S}_0}$$

となるような前記集合の三角形t_iを含む。

【００１５】

・前記残りの部分集合L^*の各三角形は、三角形が

【数２】

$$P(t_i \in \mathcal{L}^k) = \begin{cases} \frac{\mathcal{A}(t_i)}{\mathcal{S}_N}, & \text{if } k = N \\ \frac{\mathcal{A}(t_i)}{\mathcal{S}_k} - \frac{\mathcal{A}(t_i)}{\mathcal{S}_{k+1}}, & \text{if } 0 \leqslant k < N \end{cases}$$

で定義される部分集合L^k内に存在する確率に従って、残りの部分集合L^kのうちの１つに送出（Ｓ５２０）され、ここでS_kの値はｋが増加すると増加する。

【００１６】
・S_kは$S_k = S_o \mu^k$を満たす集合であり、ここでμ＞１であり、μの値は［１；５］間の実数である。
【００１７】
・S_kは$S_k = S_o \mu^k$を満たす集合であり、ここでμの値は、三次元シーンの半径であるR_{scene}を用いて、
【数３】

$$\mu > \sqrt[N]{4\frac{\mathcal{R}_{scene}^2}{S_o}}$$

で定義される。
【００１８】
・前記残りの部分集合L^*の各三角形が送出された後、前記残りの部分集合L^*の各三角形について単一の仮想点光源を算出し、ここで、確率法則を用いて前記集合の各三角形に影響半径を割り当てるステップはさらに、前記残りの部分集合L^*の各三角形の仮想点光源の出射放射輝度を算出するステップを含み、前記算出は、三角形が送出された前記部分集合に従って実行される。
【００１９】
・各三角形の仮想点光源について出射放射輝度を算出するステップは、仮想点光源の寄与度が大きい三次元シーン内の点の集合を表す入れ子状のボール群$B_h(t_i)$を算出するステップを含む。
【００２０】
・前記Ｎ＋２個の部分集合のうち、残りの部分集合のそれぞれについて、残りの部分集合上のサポート関数（ｓｕｐｐｏｒｔ　ｆｕｎｃｔｉｏｎ）の集合$(f^k)_k$が、入れ子状のボール群$B_h(t_i)$の境界に亘って一定である１の分割を形成するようなサポート関数$f^k(t_i, x)$を算出する。
【００２１】
・前記サポート関数の集合$(f^k)_k$の基底関数（ｂａｓｅ　ｆｕｎｃｔｉｏｎ）はＢスプラインである。
【００２２】
・$B_h(t_i)$は、
【数４】

$$\mathcal{B}_h(t_i) = \{x \epsilon \mathbb{R}^3 \; s.t. \frac{||x - y_i||}{\langle \overrightarrow{n_x}, \bar{x}y_i \rangle^+} \leq D(h) = \frac{1}{\pi}\sqrt{\frac{\rho_x \rho_i E(t_i)}{h}}\}$$

で定義され、ここで、ｘは法線
【数５】

$$\overrightarrow{n_x}$$

を有する三次元シーンの点であり、y_iは三角形t_i上の仮想点光源の中心となる点であり、ρ_xは点ｘにおけるアルベドであり、ρ_iは点y_iにおけるアルベドであり、$E(t_i)$は、三角形t_iへの直接放射照度である。

【００２３】
・描画ステップにおいて、入れ子状のボール群B_h（t_i）の球面を描画するときに、サポート関数の集合（f^k）$_k$が、スプラッティング演算（ｓｐｌａｔｔｉｎｇ　ｏｐｅｒａｔｉｏｎｓ）を実行するのに用いられる。
【００２４】
・前記閾値S_oは、
【数６】

$$S_o = 4\pi \frac{D_{near}^2}{N_{Avg}}$$

で定義され、ここで、N_{Avg}は、仮想点光源の平均個数であり、D_{near}は、前記３Ｄシーンの画素とＶＰＬとの間の最短距離である。
【００２５】
・前記集合の各三角形t_iが閾値S_o以下の面積を有すると判定する前に、前記集合の三角形t_iの各頂点について単一のランダムな整数を生成する（Ｓ２０）。
【００２６】
・ここで、三角形の各頂点について単一のランダムな整数を算出するステップは、さらに、前記第３の集合の三角形の各副頂点について、当該副頂点の重心座標を決定するステップと、前記重心座標を用いてノイズ・テクスチャ内のランダムな値を取得するステップ（Ｓ２１０）と、前記取得したランダムな値を用いて前記第３の集合の三角形の前記各副頂点の前記単一のランダムな整数を生成するステップ（Ｓ２２０）とを含む。
【００２７】
さらには、前記方法を実行するための命令を含むコンピュータプログラムが提供される。さらには、前記方法を実行するための命令を含む、グラフィック・ライブラリ・コンピュータプログラムが提供される。
【００２８】
さらには、前記グラフィック・ライブラリ・コンピュータプログラムを記録したコンピュータ読み取り可能記憶媒体が提供される。
【００２９】
さらには、当該コンピュータプログラムを記録したメモリに接続されたプロセッサと、グラフィック処理ユニットを有するグラフィック・カードとを備えるシステムが提供される。
【００３０】
前記システムのグラフィック・カードは、前記集合の各三角形t_iが閾値S_o以下の面積を有すると判定するジオメトリ・シェーダ・ユニットと、前記集合の三角形t_iをテッセレーションするためのテッセレーション・ユニットとを備えていてもよく、前記メモリ上に記録された前記コンピュータプログラムは、前記単一のランダムな整数の生成を実行するために前記テッセレーション・ユニットを構成するように適合されている。
【図面の簡単な説明】
【００３１】
【図１】本方法の一例のフローチャートを示す。
【図２】本方法の一例のフローチャートを示す。
【図３】本方法の一例を示す。
【図４】本発明の間接イルミネーション・シェーダ・パイプラインの一例を示す。
【図５】本発明の間接イルミネーション・パイプラインの一例を示す。
【図６】点ｘにおける仮想点光源のイルミネーションの一例を示す。
【図７】０（黒）から１（白）の範囲の値について、平面部分上のサポート関数の視覚化の一例を示す。

【図８】一次元の１の分割の一例を示す。
【図９】三角形ごとの一様なランダム値生成の一例を示す。
【図１０】本発明の非標準パイプラインを使用しない場合、および使用する場合における、同じシーンのイルミネーションの例を示すスクリーンショットである。
【図１１】本発明の非標準パイプラインを使用しない場合、および使用する場合における、同じシーンのイルミネーションの例を示すスクリーンショットである。
【図１２】本方法を実行するシステムの一例を示す。
【発明を実施するための形態】
【００３２】
図１および２のフローチャートを参照して、コンピュータによって実施される、三次元シーンのグローバル・イルミネーションを描画するための方法を提案する。本方法は、三角形の集合Ｌと１つまたは複数の直接光源とを含む三次元シーンを提供することを含む。次いで、前記集合の各三角形t_iが閾値S_o以下の面積を有すると判定する。次に、確率法則を用いて、前記集合の各三角形に影響半径を割り当てる。次いで、前記三角形を、それぞれの影響半径に応じて選別することによって、三角形の部分集合L^*を得る。次に、前記三次元シーンが、その三角形の集合へのライティングによって描画される。前記三角形の部分集合L^*の三角形がその影響半径に応じて間接光源として用いられる。
【００３３】
このような方法は、三次元シーンのグローバル・イルミネーションのリアルタイムでの描画を改善する。実際には、照明のマルチスケール表現が生成される。つまり、ＶＰＬが、入力される三角形の集合の確率的な間引き処理に基づいて生成される。そして、前方に向かって（すなわち、光源からセンサーまでの光空間を探索して）、カットで選択されたかのように、それぞれの有界影響領域を算出する。本発明は、グラフィック・カードのジオメトリ・シェーダとテッセレーション・ユニットの両方を利用して、間接照明に近似する仮想点光源（ＶＰＬ）の集合をフレーム単位で構築している。本発明は、２パス戦略においてジオメトリ駆動型ＶＰＬの集合を精緻化および単純化することができる、イルミネーション用拡散グラフィック・パイプラインを提案する。膨大な数の三角形を有するシーンの文脈において、オブジェクト空間放射輝度キャッシュの解像度を調整するために、テッセレーション・ユニットとジオメトリ・シェーダの両方を利用する。本発明は、複雑な前処理を示唆するものではなく、また、フレームにわたって複雑なデータ構造を保持する必要もない。光照射野のマルチスケール表現を使用することで、リアルタイムのパフォーマンスを実現し、それは完全に動的であり、ツリー構造に頼らず、また、フレーム間でデータ構造を維持する必要もない。これは、ライト、ビューポイント、ジオメトリ・アニメーションなどを含む、完全に動的なシーンと互換性がある。最後に、統合に関して、本発明は、当然、標準的な現代のグラフィック・パイプラインに適合し、ホスト・アプリケーションによって採用される相補的描画技術について、強い仮定は何ら必要としない。
【００３４】
本方法は、コンピュータによって実施される。すなわち、本方法のステップ（あるいは略全てのステップ）が少なくとも１つのコンピュータ、または類似の任意のシステムによって実行される。よって本方法のステップは、コンピュータにより、場合により、完全に自動的に、あるいは半自動的に実行される。例えば、本方法の少なくともいくつかのステップは、ユーザとコンピュータの対話を通じて始動されてもよい。求められるユーザとコンピュータの対話レベルは、想定される自動性のレベルに応じたものであって、ユーザの要望を実装する必要性との間でバランスをとるものとしてもよい。例えば、このレベルは、ユーザが設定し、および／または、予め定義されていてもよい。
【００３５】
本方法のコンピュータによる実施の典型的な例は、この目的に適したシステムを用いて本方法を実行することである。当該システムは、本方法を実行するための命令を含むコンピュータプログラムを記録したメモリに接続されたプロセッサ、および、グラフィック処理ユニットを有するグラフィック・カードを備えていてもよい。コンピュータプログラム

は、グラフィック・ライブラリ・コンピュータプログラムである可能性がある。また、メモリはデータベースを記憶していてもよい。メモリは、そのような記憶に適した任意のハードウェアであり、場合により、物理的に区別可能ないくつかの部分（例えば、プログラム用に１つ、場合によりデータベース用に１つ）を含む。データベースは、描画すべき３Ｄシーンを記憶していてもよく、例えば、データベースは、当該シーンの三角形の集合を記憶する。

【００３６】

図１２は、本システムの一例を示すものであって、当該システムは、クライアントコンピュータシステム、例えばユーザのワークステーションである。

【００３７】

本例のクライアントコンピュータは、内部通信バス１０００に接続された中央演算処理装置（ＣＰＵ）１０１０、および同じくバスに接続されたランダムアクセスメモリ（ＲＡＭ）１０７０とを備える。クライアントコンピュータは、さらに、バスに接続されたビデオランダムアクセスメモリ１１００と関連付けられたグラフィックス処理装置（ＧＰＵ）１１１０を備える。ビデオＲＡＭ１１００は、当該技術分野において、フレームバッファとしても知られる。大容量記憶装置コントローラ１０２０は、ハードドライブ１０３０などの大容量記憶装置へのアクセスを管理する。コンピュータプログラムの命令及びデータを具体的に実現するのに適した大容量メモリ装置は、例として、ＥＰＲＯＭ、ＥＥＰＲＯＭ及びフラッシュメモリ装置のような半導体メモリ装置、内蔵ハードディスクやリムーバブルディスクなどの磁気ディスク、光磁気ディスク、およびＣＤ－ＲＯＭディスク１０４０を含む、全ての形式の不揮発性メモリを含む。前述のいずれも、特別に設計されたＡＳＩＣ（特定用途向け集積回路）によって補完されてもよいし、組み入れられてもよい。ネットワークアダプタ１０５０は、ネットワーク１０６０へのアクセスを管理する。クライアントコンピュータはまた、カーソル制御装置、キーボードなどの触覚装置１０９０を含んでいてもよい。カーソル制御装置は、ユーザがディスプレイ１０８０上の任意の所望の位置にカーソルを選択的に位置させることを可能にするために、クライアントコンピュータ内で使用される。さらに、カーソル制御デバイスは、ユーザが様々なコマンドを選択し、制御信号を入力することを可能にする。カーソル制御装置は、システムに制御信号を入力するための多数の信号生成装置を含む。典型的には、カーソル制御装置はマウスであってもよく、マウスのボタンは信号を生成するために使用される。あるいは、または追加的に、クライアントコンピュータシステムは、感知パッドおよび／または感知スクリーンを備えてもよい。

【００３８】

コンピュータプログラムは、コンピュータによって実行可能な命令を含んでいてもよく、命令は、上記システムに方法を実行させるための手段を含む。プログラムは、システムのメモリを含む任意のデータ記憶媒体に記録可能であってもよい。プログラムは、例えば、デジタル電子回路、またはコンピュータハードウェア、ファームウェア、ソフトウェア、またはそれらの組み合わせで実装されてもよい。プログラムは、例えばプログラマブルプロセッサによる実行のための機械読み取り可能な記憶装置に具体的に実現された製品のような装置として実装されてもよい。方法ステップは、プログラム可能なプロセッサが命令のプログラムを実行し、入力データを操作して出力を生成することによって方法の機能を実行することによって実行されてもよい。したがって、プロセッサは、データ記憶システム、少なくとも１つの入力デバイス、および少なくとも１つの出力デバイスからデータおよび命令を受信し、また、それらにデータおよび命令を送信するようにプログラム可能であってもよく、またそのように接続されていてもよい。アプリケーションプログラムは、高水準の手続き型またはオブジェクト指向のプログラミング言語で、または必要に応じてアセンブリ言語または機械語で実装されていてもよい。いずれの場合も、言語はコンパイラ型言語またはインタープリタ型言語であってもよい。プログラムは、フルインストールプログラムまたは更新プログラムであってもよい。いずれの場合も、プログラムをシステムに適用すると、本方法を実行するための指示が得られる。

【００３９】
以下に、より詳細に論じるように、本発明は仮想点光源（Ｖｉｒｔｕａｌ　Ｐｏｉｎｔ　Ｌｉｇｈｔｓ：ＶＰＬｓ）、すなわち、ある光がキャッシュされる点の集合を用いる。キャッシュされる照明は、直接照明であってもよい。ＶＰＬは、間接照明の反射を近似するのに使用される。図５は、当該技術分野で知られているＶＰＬベースのパイプラインの一例を示す。ＶＰＬパイプラインは、典型的には、照明の３つの主要ステージ、すなわち（ａ）ＶＰＬ生成、（ｂ）間接ライト・キャッシング、および（ｃ）ＶＰＬを用いた照明からなる。

【００４０】
本発明は、特に、（ａ）ＶＰＬ生成、および（ｃ）照明中にそれらを使用することに焦点を合わせる。図１は、ＶＰＬの生成の対象となる三角形の生成の一例を示す。
・前処理時に、動的ジオメトリ変換の下であっても一貫して三角形ごとの乱数を生成するのに描画時に使用される、単一の頂点ごとのランダムな整数を算出する。
・生のジオメトリから三角形の部分集合を生成する一例について説明する。特に、通常のパイプラインと非標準パイプラインの使用について示す。大きな三角形（本発明においては「非標準（ｄｉｖｅｒｇｅｎｔ）」と呼ぶ）は、前述の戦略では十分にサンプリングされず、最終的な画像に、重大な照明アーチファクトをもたらす可能性がある。その結果、非標準三角形は、テッセレーション・ユニットを経由し、そこで適切な解像度に達するまで細分化される。
・生データの三角形の破棄の一例として、例えば、サンプリング情報の量を最適化しつつ、管理された三角形の数を大幅に減らすために、主に小さな三角形からなる特別な集合が完全に破棄される。

【００４１】
図２は、ＶＰＬの生成の一例を示し、ここでは、生データ（図１）から生成された三角形の完全な集合が、確率法則に従ってランダムに分割されている。各三角形に対してＶＰＬが生成され、その出射放射輝度は、関連する区画（L^0, ・・・, L^N）のみに基づいて算出される。最後に、間接イルミネーションは、典型的な遅延シェーディングプロセスで、ＶＰＬ区画に応じてスプラット機能サポートを用いてスプラットされ、強力なＶＰＬを確保することにより三日月形サポートを用いて遠隔照明を実行する。

【００４２】
図１に戻って、本発明の第１の態様、すなわち、ＶＰＬの生成対象である生データから三角形を生成することについて説明する。ステップＳ１０において、３Ｄシーンが提供される。３Ｄシーンは、三角形の集合Ｌと、１つまたは複数の直接光源とを含む。提供するとは、生データ（３Ｄシーンの三角形の集合Ｌ）が、本発明の方法を実施するシステムおよび／またはプログラムによってアクセス可能であることを意味する。典型的には、３Ｄシーンは、当該技術分野で知られているように、アプリケーションの要求に応じて表示される画像を生成するフレームワークである描画エンジンにロードすることができる。例えば、ＣＡＤシステムのＣＡＤアプリケーションは、３Ｄモデル化オブジェクトの３Ｄシーンを、入力として、描画エンジンに提供し、描画エンジンは、ＣＡＤシステムの１つまたは複数のグラフィック・カードを用いて、３Ｄシーンをスクリーンに描く。３Ｄシーンの三角形はメッシュを形成する。例えば、３Ｄシーンのモデル化オブジェクトは、当該技術分野で知られているように、互いに接続された三角形によって表される。

【００４３】
次いで、ステップＳ２０において、集合Ｌの各三角形の各頂点について単一のランダムな整数が生成される。その結果、頂点ごとのｕｉｎｔ３２属性ｖ＿ｒａｎｄが追加される。この属性は、０から2^{32}、あるいはさらに０から2^{64}までの範囲（ただしこの範囲に限定されない）で一様に選択されたランダムな値を用いてメッシュをロードするときに初期化される。なお、この工程は、当該技術分野で知られているように行うことができる。

【００４４】
ステップＳ１０およびＳ２０は、図３の（ａ）に示されている。

【００４５】
次いで、ステップＳ３０において、集合の各三角形t_iが、閾値S_o以下の面積を有すると判定される。判定の結果に応じて、生のデータは、集合の、通常の三角形および非標準三角形に分割される（または分割されない）。ここで、非標準三角形は、図３の（ｄ）および（ｅ）に示すように、すべての新しい三角形が通常の三角形であると考えられるように細分化される、閾値S_oより大きな面積を有する三角形であり、通常の三角形は、閾値S_o以下の面積を有する三角形である。ステップＳ３１０～Ｓ３１２は、通常の三角形の通常のパイプラインについて論じ、ステップＳ３２０～Ｓ３２２は、非標準パイプラインについて論じている。
【００４６】
通常の三角形は、非標準三角形、すなわち、ある閾値Ｓｏより大きい表面積Ａ（t_i）を有する三角形t_iの集合と区別される。この閾値は、（式１）
【数７】

$$S_o = 4\pi \frac{D_{near}^2}{N_{Avg}}$$

となるように設定することができ、これは、画素から少なくともD_{near}だけ離れているN_{Avg}個のＶＰＬを用いて画素をほぼ点灯させることを目的とするヒューリスティックである。D_{near}は、画素とＶＰＬとの間の最短距離を提供する変数である。N_{Avg}は、３Ｄシーン上のＶＰＬの個数である。R_{scene}をシーン半径とすると、D_{near}はD_{near}＝０．２×R_{scene}と設定することが可能で、N_{Avg}は、所望の品質／スピードのトレードオフに応じて、好ましくは、６４と１０２４の間に設定される。なお、N_{Avg}の値の範囲は変更可能であり、ＶＰＬの個数は、描画を実行するシステムの演算リソースに応じて、より多くなり得る。シーンの直径は、$Diam_{scene}$＝２×R_{scene}と設定され、$Diam_{scene}$は、シーンの２点間の最長距離である。
【００４７】
したがって、ステップＳ３００において、三角形t_iの集合は、S_o以下の面積を有する三角形の第１の集合（Ｓ３１０）およびS_oよりも大きい面積を有する三角形の第２の集合（Ｓ３２０）に分割される。なお、第１の集合が、S_o未満の三角形を含み、第２の集合が、S_o以上の三角形を含んでいてもよい。
【００４８】
ステップＳ３１２において、第１の集合の三角形が、例えばキャッシュ・メモリに格納される。
【００４９】
ステップＳ３２２において、第２の集合の三角形は、三角形の第３の集合へとテッセレーションされる。テッセレーションは、当該技術分野で知られているように実行される。テッセレーションされた三角形は、S_o以下の面積を有する。すなわち第３の集合の各三角形は、テッセレーションの結果として、閾値以下の面積を有する。
【００５０】
ここで、生データは、三角形の２つの集合からなる。これらの集合の三角形は、共にS_o以下の面積を有する。これら集合の違いは、第１の集合の各三角形の各頂点は単一のランダムな整数に関連付けられているが、第２の集合の三角形はそうではないという点であり、これは、テッセレーションは、１つまたは複数の頂点が以前に生成された単一のランダムな整数を有していない、新たな三角形を作成するためである。図１の例では、単一のランダムな整数が生成され、生データに関連付けられている。なお、単一のランダムな整数は、後のステージで生成して関連付けてもよい。例えば、第１および第２の集合が決定されている間や、生データの分割が実行された後であってもよい。例えば、第１の集合が格納されると、第１の集合の三角形の頂点について単一のランダムな整数が生成されるかもしれない。

【００５１】
次いで、ステップＳ２００～Ｓ２２０は、第２の集合の各三角形の各頂点について単一のランダムな整数を生成する一例を提供する。好ましくは、これらのステップは、第２の集合の三角形をテッセレーションする間に実行される。第２の集合の、新たにテッセレーションされた三角形ごとに（Ｓ３２０）、ステップＳ２００～Ｓ２２０が実行される。あるいは、これらは、第３の集合が構築された後に実行されてもよい。ここで、ステップＳ２００～Ｓ２２０について、第２の集合の三角形がテッセレーションされるたびに生成が行われる場合について説明する。
【００５２】
ステップＳ２００において、第３の集合の三角形の各副頂点を決定する。図９の（ａ）に示すように、第２の集合の三角形の各頂点は、ｕｉｎｔ３２属性、すなわちｖ＿ｒａｎｄを含む。図９の（ｂ）において、三角形９０の副頂点９０ａ、９０ｂ、９０ｃが得られる。なお、三角形９０は図９の（ａ）の三角形から得られた新しい１０個の三角形のうちの１つであることが理解される。
【００５３】
次いで、ステップＳ２１０において、第３の集合の三角形の上記各副頂点の重心座標ｔｅｓｓ＿ｃｏｏｒｄ［ｉ］を特定する。重心座標は、第３の集合の三角形の上記各副頂点を得た第２の集合の三角形のテッセレーションの間に算出される。次いで、重心座標ｔｅｓｓ＿ｃｏｏｒｄ［ｉ］を用いて、ノイズ・テクスチャ内でランダムな値が取得される。より正確には、各ｔｅｓｓ＿ｃｏｏｒｄ［ｉ］について、ランダムな値が取得される。この取得は、当該技術分野で知られているように実行される。すなわち、テクスチャは、テーブルの各セルが値を格納する二次元のテーブルと考えることができ、例えば、値は３２ビットの整数である。ランダムな値が取得されると、セルの１つが読み取られ、座標ｔｅｓｓ＿ｃｏｏｒｄ［ｉ］を用いて、テーブル内のセルに到達する。ノイズ・テクスチャは、一般に、予め、例えば、３Ｄシーンの描画が要求されたときに算出される。ノイズ・テクスチャは、例えば１０２４×１０２４など、所定のサイズのテーブルである。
【００５４】
次に、ステップＳ２２０において、取得されたランダムな値を用いて、第３の集合の三角形の各副頂点について、単一のランダムな整数を生成する。図９の（ｃ）に示すように、三角形９０の各頂点は、テッセレーションされた三角形上の頂点の重心座標ｔｅｓｓ＿ｃｏｏｒｄ［ｉ］から生成されたランダムな値ｖ＿ｒａｎｄ＿ｔｅｓｓ［１］に関連付けられる。
【００５５】
第３の集合の三角形のいくつかは、ステップＳ２０で生成されたランダムな値にすでに関連付けられている。これらのランダム値は保持され、ランダム値を持たない副頂点のみがステップＳ２００～Ｓ２２０で考慮される。このように、単一のランダムな整数を持たない第３の集合の三角形の副頂点v_iを決定する。これにより、演算リソースを保存できる。
【００５６】
ここで、第１および第３の集合は、S_o以下の面積を有し、頂点が単一のランダム値に関連付けられた三角形を有する。第１および第３の集合は、非標準パイプラインが、非標準三角形を通常の三角形に変換するときに、マージすることができる（Ｓ４０）。
ここで、図２を参照して、確率法則に従ったＶＰＬの生成、およびマージされた第１および第３の集合をランダムに分割する方法について説明する。
【００５７】
ステップＳ５０において、三角形の集合（Ｓ４０）は、確率法則に従って三角形のＮ＋２個の部分集合に分割され、これらのＮ＋２個の部分集合のうちの１つは破棄される（Ｓ５１０）。すなわち、三角形が確実に除去される。
【００５８】
実際、ステップＳ５１０における三角形の間引きは、分割する前に行うことができるの

で、分割する際に分析すべき三角形が少なくて済み、区画を作成するために必要な演算リソースが少なくて済む。

【００５９】

三角形の間引きは、３Ｄシーンの小さな三角形を削除することを目的とする。これは、それらが、拡散した間接照明の最終的な描画に、あまり寄与しないためである。単純な解決策としては、面積が閾値以下の小さい三角形を除去することが考えられる。しかしながら、小さな三角形のグループは、全体として、光輸送に重要な影響を及ぼすことがある。確率的な間引きアプローチが用いられ、このアプローチでは、第１および第３の集合の各三角形に対して、０と１との間にある均一なランダム値u_{ti}を算出することにより、強くテッセレーションされたジオメトリの寄与度が保持される。そのルールは次の通りである。すなわち、$A(t_i) > u_{ti} S_o$の場合、三角形を保持し（すなわち、破棄しない）、$A(t_i) \leqq u_{ti} S_o$の場合、三角形を、破棄する三角形の集合に追加する（なお、三角形は即座に破棄されてもよい）。

【００６０】

すべての三角形が通常の三角形であると仮定できるとすると、三角形t_iが維持される確率は以下のようになる（式２）。

【数８】

$$\forall t_i \in \mathcal{L},\ P(t_i \in \mathcal{L}^*) = \frac{\mathcal{A}(t_i)}{\mathcal{S}_0}$$

ここで、L^*は、保持されている第１および第３の集合の三角形t_iを示し（Ｓ４０）、Ｌは第１および第３の集合の三角形を示す。直観的には、これは、三角形が小さいほど、破棄される可能性が高くなることを意味する。それと同時に、この分割は、シーン全体の表面上のサンプルの均一な分布に変換され、その結果、保持された三角形の数の期待値は、A_{scene}をシーン全体の面積として、

【数９】

$$\mathbb{E}(N_{sample}) = \frac{\mathcal{A}_{scene}}{\mathcal{S}_o}$$

となる。

【００６１】

三角形が破棄されるか、または少なくとも破棄される区画に関連付けられると、残りの三角形（すなわち、保持される三角形）が、三角形が部分集合L^k内に存在する確率に従って、Ｎ＋１個の残りの部分集合L^kのうちの１つ（L^0，・・・，L^N）に送出される（ステップＳ５２０）。送出は、L^Nが少数の三角形を含み、ｋが０に近づくほどL^kがより密集するように行われる。そのとき代表的なシーンの三角形のマルチスケール分割は、間引き戦略（ステップＳ５１０）から現れ、三角形が部分集合L^k内にある確率は（式３）

【数１０】

$$P(t_i \in \mathcal{L}^k) = \begin{cases} \dfrac{\mathcal{A}(t_i)}{\mathcal{S}_N}, & \text{if } k = N \\ \dfrac{\mathcal{A}(t_i)}{\mathcal{S}_k} - \dfrac{\mathcal{A}(t_i)}{\mathcal{S}_{k+1}}, & \text{if } 0 \leqslant k < N \end{cases}$$

であり、$\mu > 1$をユーザ定義による実数とすると、（式４）

【数１１】

$$S_k = S_o \mu^k$$

である。$S_k = S_o \mu^k$では、部分集合L^k当たりのＶＰＬの数は指数関数的に増加する。これは先行技術のいくつかの既存技術、例えばツリー構造のものの挙動に似ている。

【００６２】

μの値は、シーンと区画数によって異なる。μの値は、典型的には１、４と５の間である。μの許容値の要件については、本方法の実施の一例についての議論において後述する。実際、$S_k = S_o \mu^k$は実施上の選択肢であり、区画（または部分集合）L^kに存在する三角形の平均表面として理解してもよい。実際には、S_kは、ｋが増加すると、その値が増加するように選択される。上述の式（式４）により、分割は、光照射野で使用される従来の階層表現に似た、ジオメトリ的に減少するサイズ期待値を有する部分集合が生じる。ただし、三角形は他の選択肢とは独立して特定の部分集合の影響を受けるため、維持、生成、あるいは管理されるような明示的な階層は存在しない。特に、これは、所与の部分集合内に位置する所与の三角形が、より細かい三角形によって運ばれる粗い情報を捕捉しないことを意味する。

【００６３】

以下のアルゴリズム１は、確率法則u_{ti}を変更せずに、式３から定義されるような区画の生成の一例を示す。

【数１２】

```
1: procedure COMPUTELEVEL(u_ti ; ti)
2:   for k ← N ... 1 do
3:       if u_ti < A(t_i)/S_k then
4:           return k
5:       end if
6:   end for
7:   DISCARDTRIANGLE( )fthreshol
```

【００６４】

各三角形についてアルゴリズム１を評価するために、変数u_{ti}に擬似ランダム値が使用される。一例では、この値の生成は、追加の頂点単位のｕｉｎｔ３２属性ｖ　ｒａｎｄに依存する。上述のとおり、この属性は、例えば０と2^{32}の間で一様に選択されたランダムな値を用いてメッシュ（生データ）をロードするときに初期化される。通常のパイプラインでは、ジオメトリシェーディングステージで、u_{ti}が三角形ｔｉの３つの頂点のランダム属性間のｘｏｒとして算出される。ｘｏｒ演算子を選択することにより、２つまたは３つの三角形の間の相関を避けることができる。さらに、新しいランダム値は、描画時にフレームごとに最大１回更新可能なグローバルで一様なランダム値であるｕ＿ｒａｎｄを用いることによって、いつでも生成することができる。元の頂点ごとの値でｘｏｒされているので、メッシュ上にまったく新しいランダム分布を得ることができる。これは以下のアルゴリズム２で明らかである。

【００６５】

非標準パイプライン（通常の三角形ではない三角形のためのもの）では、u_{ti}の構築は、さらに非常に微妙である。実際、この値は、カメラが動いたりメッシュが変形したりしても、すなわちメッシュトポロジが同じままである限り、変更されないままである。これらの特性を保持するために、モデル空間でテッセレーションを実行し、テッセレーション

パターンが入力三角形の形状にのみ依存するという事実を利用する。実際に、テッセレーションパラメータは、ステップＳ２００～Ｓ２２０に関する議論で説明したように、元の三角形領域によってのみ決定される。したがって、細分化から出現する三角形についてu_{ti}を生成するために、アルゴリズム２に示すように、３つのｖ＿ｒａｎｄ＿ｔｅｓｓ、３つの基本三角形ｖ＿ｒａｎｄ、および最終的にグローバルで一様な乱数ｕ　ｒａｎｄの間でｘｏｒが算出される。

【数１３】

```
1: uniform uint32 u rand
2: uniform texture2D noise
3: function REGULARRAND(ivec3 v rand)
4:     return u rand xor v rand.x xor v rand.y xor v rand.z
5: end function
6: function DIVERGENTRAND(ivec3 v rand, vec2 tess coord[3])
7:     ivec3 v tess rand
8:     v rand tess.x = TEXTURE(noise, tess coord[0])
9:     v rand tess.y = TEXTURE(noise, tess coord[1])
10:    v rand tess.z = TEXTURE(noise, tess coord[2])
11:    return REGULARRAND(v rand xor v tess rand)
12: end function
```

【００６６】

図３の（ｂ）および（ｃ）は、区画の生成、すなわち、生き残っている三角形（「小さい」三角形（ｂ）の破棄後に残っているもの）を、変数u_{ti}に使用される擬似ランダム値を用いて、式３の確率法則

【数１４】

$$P(t_i) \epsilon \mathcal{L}^k$$

に従って、部分集合L^0、・・・L^k、・・・L^Nに分類する。

【００６７】

図２に戻って、ステップＳ６０で、残りの部分集合L^*の各三角形について、単一の仮想点光源が算出される。好ましくは、ハードウェアの制約に適切に対処するために、１つの三角形当たり１つのＶＰＬしか算出されない。なお、いくつかの三角形は複数のＶＰＬを有する。例えば、決定はそれらのＡ（t_i）に従って行うことができる。残りの三角形それぞれについてのＶＰＬの算出は、当該技術分野で知られているように実行される。

【００６８】

ＶＰＬの算出は、実際には、ＶＰＬを用いて照明を行うために、各ＶＰＬについての出射放射輝度の算出を含む。以下の数学的議論で明らかなように、出射放射輝度の算出（ステップＳ７０）は、三角形が送出された部分集合に従って実行される。したがって、影響半径（すなわち、出射放射輝度）は、式３の、部分集合L^0、・・・L^k、・・・L^Nを得るために用いられる確率法則を用いて、第１および第３の集合の各三角形に割り当てられる。したがって、確率法則は、影響半径の密度が減少するように実行される、影響半径の分布と考えることができる。実際、分割により、ジオメトリ的に減少するサイズ期待値を有する部分集合が生じ、これは光照射野で用いられる従来の階層表現に似ている。なお、

３Ｄシーンの描画にあまり寄与しない三角形を破棄することは、その影響半径に応じて三角形を選別することにより、三角形の初期集合（生データ）の部分集合を得ることに相当する。

【００６９】

ステップＳ７００～７１０は、三角形が送出された部分集合L^*に従って、L^*の各三角形の仮想点光源についての出射放射輝度を算出する例を示す。入れ子状のボール群B_h（t_i）が算出される。B_h（t_i）は、仮想点光源の寄与度が大きい三次元シーン内の点の集合を表している（Ｓ７００）。ステップＳ７１０において、残りのＮ＋１個の部分集合のそれぞれについて、サポート関数f^k（t_i，ｘ）が算出される。残りの部分集合に対するサポート関数の集合$(f^k)_k$は、入れ子状のボール群B_h（t_i）の境界上に亘って一定である１の分割を形成する。説明のために、この例で使用される出射放射輝度のモデルがここで論議され、ステップＳ７００およびＳ７１０の演算がそれに応じて実行される。図３の（ｆ）は、異なるサイズの影響半径の例を示す。三角形の仮想点光源について算出される出射放射輝度は、三角形が属する部分集合に依存する。

【００７０】

法線n_xを有する点ｘの間接出射放射輝度

【数１５】

$$L(x,\overrightarrow{n_x})$$

は、式５によって次のように定義される。

【数１６】

$$L(x,\overrightarrow{n_x})=\sum_{t_i\epsilon\mathcal{L}} H(t_i,x,\overrightarrow{n_x})\mathcal{A}(t_i)$$

【００７１】

ここでＬは、シーン内のすべての三角形の集合（すなわち生データ）を表し、

【数１７】

$$H(t_i,x,\overrightarrow{n_x})$$

は、

【数１８】

$$\overrightarrow{n_x}$$

によって方向づけられた、t_iから始まりレシーバｘへと向かう入射放射輝度伝達関数を表す。図６は、

【数１９】

$$\overrightarrow{n_x}$$

によって方向づけられた点ｘを有する画素６０（レシーバ）を示し、ρ_xは、点ｘにおけ

るアルベドである。アルベドρ_xを用いてレシーバを拡散させるために、この伝達関数は次のように定義される（式６）。

【数２０】

$$H(t_i, x, \vec{n}_x) = L(t_i, y_i\bar{}x)\frac{\rho_x}{\pi}\frac{\langle \vec{n_x}, x\bar{}y_i\rangle^+ \langle \vec{n_i}, y_i\bar{}x\rangle^+}{d_i^2}$$

【００７２】

ここで、

【数２１】

$$\bar{u} = \frac{\vec{u}}{\|\vec{u}\|}, \langle \vec{u}, \vec{v}\rangle^+ = \max(0, \langle \vec{u}, \vec{v}\rangle), L(t_i, y_i\bar{}x)$$

は、

【数２２】

$$y_i \epsilon t_i$$

を中心とするＶＰＬから

【数２３】

$$\bar{x}y_i$$

の方向へと向かう放射輝度であり、

【数２４】

$$d_i = max(\epsilon, \| \overrightarrow{xy_i} \|)$$

は、特異点を避けるためにユーザパラメータεに固定されたy_iとｘとの間の距離である。典型的には、εは、［10^{-3}；10^{-6}］の範囲に含まれる値を有する。

【００７３】

光の第１の拡散反射は、以下のＶＰＬ出射放射輝度式（式７）を用いてモデル化することができる。

【数２５】

$$L(x, \overrightarrow{n_x}) = \frac{E(t_i)\rho_i}{\pi}\langle \overrightarrow{n_i}, \overline{y_i x}\rangle^+$$

【００７４】

ここで、Ｅ（t_i）は、三角形t_iへの直接放射照度である。なお、

【数２６】

$$\langle \overrightarrow{n_i}, \overline{y_i x}\rangle^+$$

の項のために、レフレクタは、もはや完璧なランバート反射とはみなすことができない。したがって、光はジオメトリの法線の方向に反射されることが好ましいが、実験は顕著な変化が現れないことを示しており、これは後で説明するように、その後の演算を大幅に軽減する。

【００７５】
ハードウェアの制約に対処するために、
【数２７】

$$L(x,\overrightarrow{n_x})$$

の算出は、ＶＰＬの部分集合の寄与度を合計することによって、すなわち、部分集合L^kの三角形のＶＰＬの寄与度を合計することによって近似することができる。したがって、
【数２８】

$$L(x,\overrightarrow{n_x})$$

の
【数２９】

$$K(x,\overrightarrow{n_x})$$

で表される推定量が定義できる（式８）。
【数３０】

$$K(x,\overrightarrow{n_x}) = \sum_{k=0}^{N} \sum_{t_i \in \mathcal{L}} H(t_i,\, x, \overrightarrow{n_x})\, F^k(t_i, x)$$

【００７６】
ここでF^k（t_i，ｘ）は、位置ｘ、エミッタt_i、およびインデックスｋの未知関数である。その式は、以下のとおりである。
【００７７】
可能性のあるすべての区画（L^0，・・・L^k，・・・L^N）の集合に対する
【数３１】

$$K(x,\overrightarrow{n_x})$$

の期待値を算出することにより、以下の式９が得られる。
【数３２】

$$\begin{aligned}
\mathbb{E}\left[K(x,\vec{n}_x)\right] &= \mathbb{E}\left[\sum_{k=0}^{N} \sum_{t_i \in \mathcal{L}^k} H(t_i^k, x, \vec{n}_x) F^k(t_i^k, x)\right] \\
&= \sum_{t_i \in \mathcal{L}} H(t_i, x, \vec{n}_x) \mathbb{E}\left[\sum_{k=0}^{N} F^k(t_i, x) \mathbb{1}_{[t_i \in \mathcal{L}^k]}\right] \\
&= \sum_{t_i \in \mathcal{L}} H(t_i, x, \vec{n}_x) \sum_{k=0}^{N} F^k(t_i, x) P(t_i \in \mathcal{L}^k),
\end{aligned}$$

ここで、

【数３３】

$$\mathbb{1}_{[t_i \in \mathcal{L}^k]}$$

は、

【数３４】

$$t_i \epsilon \mathcal{L}^k$$

の場合１に等しく、その他の場合０に等しい、インジケータ関数である。

【００７８】

仮に

【数３５】

$$K(x, \overrightarrow{n_x})$$

が、入射放射輝度

【数３６】

$$L(x, \overrightarrow{n_x})$$

の不偏推定量を表すようにしたいのであれば、関数F^k上で、次の関数方程式（式１０）を検証する必要がある。

【数３７】

$$\forall x, \sum_k F^k(t_i, x)\ P(t_i \epsilon \mathcal{L}^k) = \mathcal{A}(t_i)$$

【００７９】

式３で確立されたＶＰＬ分割戦略によれば、F^kは、次のように定義される（式１１）。

【数３８】

$$F^k(t_i, x) = \begin{cases} \mathcal{S}_0 \mu^N f^N(t_i, x), & \text{if } k = N \\ \mathcal{S}_0 \frac{\mu^{k+1}}{\mu - 1} f^k(t_i, x), & \text{if } 0 \leqslant k < N \end{cases}$$

【００８０】

式１１は、式１０の不偏条件を、１の分割の問題として書き換えることを可能にする。１の分割は、集合Ｘ内のあらゆる点ｘについてｘにおけるすべての関数値の合計が１であるような、集合Ｘから単位区間［０，１］への関数の集合である。これは、（式１２）

【数３９】

$$\forall x, \sum_k f^k(t_i, x) = 1$$

が成り立つような関数の集合$(f^k)_k$を求めることにつながる。

【００８１】

このように、ＶＰＬを用いた照明は、三角形を部分集合（L^0、・・・L^k、・・・L^N）へと分割することにより近似されている。すなわち、三角形をＮ＋１個の部分集合へと

送出するために用いられる確率法則を用いて近似されている。この近似はまた、関数の集合（f^k）$_k$を見出すことにもつながり、この集合における各関数は、部分集合のＶＰＬの入射放射輝度をサポートする。関数の台（ｓｕｐｐｏｒｔ）は、その関数の値がゼロでない点の集合である。このように、各部分集合（L^0、・・・L^k、・・・L^N）について、それが属する台に依存するサポート関数が算出される。例えば、式１７～１８に示すように、関数f^kの台は

【数４０】

$$\mathcal{B}_{D_{k+1}}$$

である。これは、f^kの台が、$h = D_{k+1}$であるような特定のｈについての関数B_hの１つであることを意味する。

【００８２】

ＣＨＲＩＳＴＥＮＳＥＮ，Ｐ．２００８．Ｐｏｉｎｔ－ｂａｓｅｄ ａｐｐｒｏｘｉｍａｔｅ ｃｏｌｏｒ ｂｌｅｅｄｉｎｇ．Ｐｉｘａｒ Ｔｅｃｈｎｉｃａｌ Ｎｏｔｅｓ ２，５，６．で議論されているＰＢＧＩツリーカット戦略からインスピレーションを受け、入れ子状のボール群B_h（t_i）が導入される。B_h（t_i）関数は$h > 0$によって特徴づけられ、所与のｈに対して、B_h（t_i）は、ＶＰＬの寄与度が大きい点の集合を表す。言い換えれば、入れ子状のボール群B_h（t_i）は、３Ｄ空間内でボリュームの集合を形成し、各$h > 0$について、単一のボリュームB_hが存在する。B_hのジオメトリ形状は、中心が、線（y_i、n_i）に拘束され（および通過し）、かつ直径Ｄ（ｈ）を有する球面である。ｈはB_hのパラメータである。B_h（t_i）の外側の各点ｘについて、伝達関数

【数４１】

$$H(t_i, x, \overrightarrow{n_x})$$

（式６、８、９を参照して説明）は、レシーバ

【数４２】

$$\overrightarrow{n_x}$$

の向きに関わらず、ｈより小さい。これは式１３に表されている。

【数４３】

$$\forall h \in \mathbb{R}^*, \quad \mathcal{B}_h(t_i) = \left\{ x \in \mathbb{R}^3 \text{ s.t. } \max_{\vec{n}} H(t_i, x, \vec{n}) \geq h \right\}$$

【００８３】

さらに、伝達関数

【数４４】

$$H(t_i, x, \overrightarrow{n_x})$$

（式６、８、９を参照して説明）は、レシーバがエミッタの正面を向いているとき、言い換えれば、

【数４５】

$$\overrightarrow{n_x} = \bar{x}y_i$$

のとき最大である。したがって、ＶＰＬ放射輝度分布モデルでは、$B_h(t_i)$関数を以下のように書くことができる（式１４）。

【数４６】

$$\mathcal{B}_h(t_i) = \{x \epsilon \mathbb{R}^3 \; s.t. \frac{||x-y_i||}{\langle \overrightarrow{n_x}, \bar{x}y_i \rangle^+} \leq D(h) = \frac{1}{\pi}\sqrt{\frac{\rho_x \rho_i E(t_i)}{h}}\}$$

【００８４】

ここで、Ｄ（ｈ）（$B_h(t_i)$の球面の直径）は、所与の部分集合に属するＶＰＬの直接出射放射輝度の寄与度の最大距離を表す。Ｄ（ｈ）はｘに依存しないため、

【数４７】

$$(\mathcal{B}_h(t_i))_{h \epsilon \mathbb{R}^*}$$

は、境界がy_iを有し、中心が線（y_i，n_i）上にある、入れ子状のボール群である。

【００８５】

３Ｄにおける１の分割$(f^k)_k$は$B_h(t_i)$の境界の球面上で一定であり得る。$B_h(t_i)$関数は、サポート関数の分離レベル（ｉｓｏｌｅｖｅｌ）である。次いで、

【数４８】

$$\mathbb{R}^3$$

から

【数４９】

$$\mathbb{R}$$

への以下のマッピング（式１５）

【数５０】

$$\forall x \epsilon \mathbb{R}^3, \qquad d(t_i, x) = \frac{||\bar{x}y_i||^2}{\langle \overrightarrow{n_x}, \bar{x}y_i \rangle^+}$$

を定義することにより、取得する１Ｄにおける１の分割

【数５１】

$$(\tilde{f}^k)_k$$

が次のように定義される（式１６）。

【数５２】

$$\forall x \epsilon \mathbb{R}^3, \; f^k(t_i, x) = \tilde{f}^k(d(t_i, x))$$

当該分野で知られているように、１の分割として、$(f^k)_k$が、スプラッティング演算を実行するのに用いられる。したがって、各画素についてＶＰＬの照明が算出されるとき、各画素上でテクスチャを混合するために各画素の評価が実行される。描画の間、１の分割$(f^k)_k$は、算出を容易にしつつ可能な限り滑らかに作られ、それらを以下のように定義する（式１７）。

【数５３】

$$\forall d \in \mathbb{R},\quad \tilde{f}^k(d) = \begin{cases} 1 & \text{if } k = 0 \text{ and } d \in]0, D_1] \\ \frac{d - D_{k-1}}{D_k - D_{k-1}} & \text{if } k > 0 \text{ and } d \in]D_{k-1}, D_k] \\ \frac{D_{k+1} - d}{D_{k+1} - D_k} & \text{if } k > 0 \text{ and } d \in]D_k, D_{k+1}] \\ 0 & \text{otherwise} \end{cases}$$

【００８６】

また、コンパクトな台を有する連続した区分的アフィン関数の集合が得られるように、（式１８）

【数５４】

$$\begin{cases} D_k = \sqrt{S_0 \mu^k}, \ \forall k \in [0 \ldots N] \\ D_{N+1} = D_N \end{cases}$$

を定義することが可能である。このように定義された関数は、D_Nがシーンの直径より大きい場合にのみ適しており、そうでない場合、シーン内に、光輸送に寄与できない点のペアが存在する可能性がある。これは、$D_N > R_{scene}$であることを意味する。これはμ（式４）についての以下の条件に変換される。

【数５５】

$$\mu > \sqrt[N]{4 \frac{\mathcal{R}_{scene}^2}{S_o}}$$

【００８７】

ここで、図７を参照して、平面部分上の$B_h(t_i)$関数をサポートするサポート関数$(f^k)_k$の視覚化の例について説明する。$B_h(t_i)$関数は、サポート関数の分離レベルであり、サポート関数の輪郭を定義する。興味深いことに、そのサポート関数上の$B_h(t_i)$の球面の寄与度の距離（D_0，D_1，D_3）は部分集合（L^0，・・・L^k，・・・L^N）に依存するため、ＶＰＬのパワーはそれが属する部分集合に依存する。したがって、ｋが０に近づくにつれて、L^kを占有する三角形（したがってＶＰＬ）の数が増加し、パワーが減少する。一方、ｋがＮに近づくにつれて、L^kが含む三角形（したがってＶＰＬ）の数が減少し、パワーが増加する。図３の（ｆ）に示すように、ローパワーＶＰＬを使用してローカル・ライティングをキャプチャし、ハイパワーＶＰＬを使用して三日月形サポート関数によるモデル遠隔照明をキャプチャする。

【００８８】

図８は、$B_h(t_i)$の球面を描画する際にスプラッティング演算を実行するのに用いられる一次元の１の分割の例を示す。関数が平滑であればあるほど、演算結果はより良好なものとなる。図８の

【数５６】

$$\tilde{f}^k$$

の例は、限られた数の重複しかなく、良好な結果となっており（横座標には、値がゼロではない関数が最大で２つある）、関数は連続的であり、凹状である。

【数５７】

$$\tilde{f}^0$$

は、星を付した曲線で表され、

【数５８】

$$\tilde{f}^1$$

は、三角形を付した曲線で、

【数５９】

$$\tilde{f}^2$$

は、点を付した曲線で、

【数６０】

$$\tilde{f}^3$$

は、ひし形を付した曲線で、それぞれ表される。いくつかのクラスの関数を用いることができる。特に、サポート関数の集合$(f^k)_k$の、良好な基底関数は、Ｂスプラインである。

【００８９】

最後に、ステップＳ８０において、三次元シーンが、その三角形の集合へのライティングによって描画される。三角形の部分集合L^*の三角形がその影響半径に応じて間接光源として用いられる。上述のように、サポート関数$(f^k)_k$は、入れ子状のボール群の$B_h(t_i)$関数の球面を描画するためのスプラッティング関数を実行するのに用いられる。描画は、当該技術分野で知られているように実行され、算出されたＶＰＬを利用するために３Ｄシーンを備えた１つまたは複数の直接光源を用いる。

【００９０】

図１０および図１１は、グローバル・イルミネーションで描画された３Ｄシーンの画像（スクリーンショット）である。図１０は、非標準パイプラインを用いずに描画されたシーンを示しており、強くテッセレーションされたジオメトリのみが間接照明を投影している。図１１は、非標準パイプラインを用いて描画された同じ３Ｄシーンを示しており、地面（４つの三角形からなる）の反射光が、シーンの多くの部分を明るみに出している。したがって、最も大きな三角形は、遠隔照明をモデル化するために用いられるL^Nに近い部分集合（群）内のものである。

【００９１】

完全を期すため、実施の詳細について、図４を参照して説明する。パイプラインには、シーン全体の、２つのジオメトリ描画パス、すなわちＧ－バッファを生成するものと、ＶＰＬを生成してスプラットするもののみが含まれている。これらは通常のパイプラインである。この生のジオメトリの適度な使用は、例えばＣＡＤ（Ｃｏｍｐｕｔｅｒ－Ａｉｄｅｄ　Ｄｅｓｉｇｎ）などのソフトウェアアプリケーションのような、膨大な数のポリゴンを必要とするアプリケーションにとって重要なメトリックである。典型的なＣＡＤシーンは、各オブジェクトが数百または数千、さらにはそれ以上の三角形を含む、数千の３Ｄモデル化オブジェクトを含む。実際には、第三のジオメトリ描画パスが発生するが、シーンのほんの一部のみ、すなわち、非標準パイプラインで処理される非標準三角形がこれに関与する。相違（ｄｉｖｅｒｇｅｎｃｅ）の基準は、非標準三角形の数が少なく留まるよう

に設定されるため、この最後のパスは、演算コストが法外に高いということはない。実際、通常のパイプラインは、フラグメント単位や頂点単位ではなく、三角形単位で演算を実行するため、ジオメトリ・シェーダ・ステージを集中的に利用する。興味深いことに、最近のＧＰＵアーキテクチャでは、法外に大きかったジオメトリ・シェーダ・ステージのオーバーヘッドが大幅に削減され、より大きなストリームのポリゴン単位の演算が可能になった。

【００９２】

ハードウェア・テッセレーション・ユニットを利用する非標準パイプラインは、大量の三角形の集合を入力として管理するようには設計されておらず、三角形が細分化を必要としない場合であっても、三角形を処理する際に、顕著なオーバーヘッドを引き起こす。同時に、これらのアーキテクチャのジオメトリステージは、ポリゴンを破棄または通過させるとき、すなわちジオメトリの増幅が必須でないときに、非常に効率的である。通常のパイプラインの第１のパスでは、シーンジオメトリ（生データ）全体が処理されるが、テッセレーションのステージは無効になる。非標準三角形（Ｓ３２０）は、ジオメトリステージで検出され、通常の三角形が確率的にサンプリングされる間に、別のバッファに格納される（Ｓ５２０）。多数のマテリアルとテクスチャを有するシーンを管理するために、三角形ごとのマテリアルインデックスを格納し、後続のパスで、マテリアルまたはテクスチャ・アトラスから情報を取得するのに使用してもよい。

【００９３】

グラフィック・ライブラリは、描画演算を最適化する機能を提供する。例えば、ＯｐｅｎＧＬの「ｇｌＤｒａｗＡｒｒａｙＩｎｄｉｒｅｃｔ」機能を用いれば、通常のものではない三角形が格納されている非標準バッファが、以降、ＣＰＵ同期なしで、第２のパスの入力ジオメトリとして直接使用される。テッセレーションステージは、この特定のパスについてのみ起動され、それらの面積が通常のパイプラインによって処理されるのに十分に小さくなるように三角形を細分化する（Ｓ３２２）のに使用される。一般に、大きな三角形の数は、三角形の合計数に対し、比較的小さい。その結果、非標準バッファの充填および頂点処理によって引き起こされるオーバーヘッドは無視できる。

ＶＰＬサポート関数$(f^k)_k$はできるだけ滑らかにすることができる。これにより、各フレームにおいて、高周波アーチファクトのない視覚的に妥当な描画をもたらすことができる。しかしながら、上述のグローバルで一様な変数ｕ＿ｒａｎｄを用いると、同じシーンについて独立した多くの描画を生成することは容易である。さらに、各三角形について、ｕ＿ｒａｎｄから２つの新しい独立した乱数値を簡単に導き出すことができる。これらの値は、式６で定義されたＶＰＬの中心y_iを三角形t_iに対して変動させるのに用いられる。

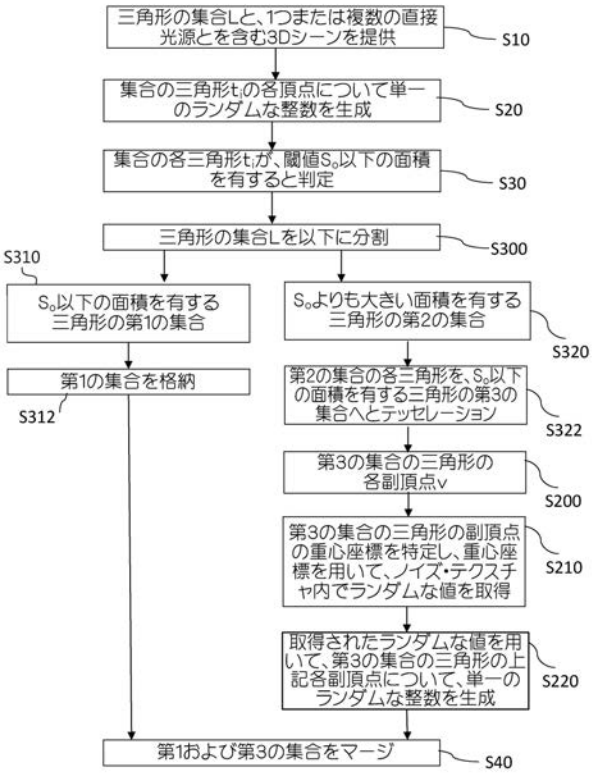
次いで、変動するＶＰＬを用いてこれらの描画のすべてを平均化することにより、式９は、拡散リフレクタで構成されるサーフェス上での第１の反射出射放射輝度の算出を可能にする結果を提供し、間接視界を無視する。これは間接照明が、あらゆるシーンの三角形上への統合によって算出されることを意味するが、本方法のこの進歩的なバージョンでは、任意の種類の乱れたジオメトリ（例えば、通常のマッピングやアルファテストされたもの）を管理することができ、また、放射テクスチャジオメトリを管理するために拡張することもできる。

【００９４】

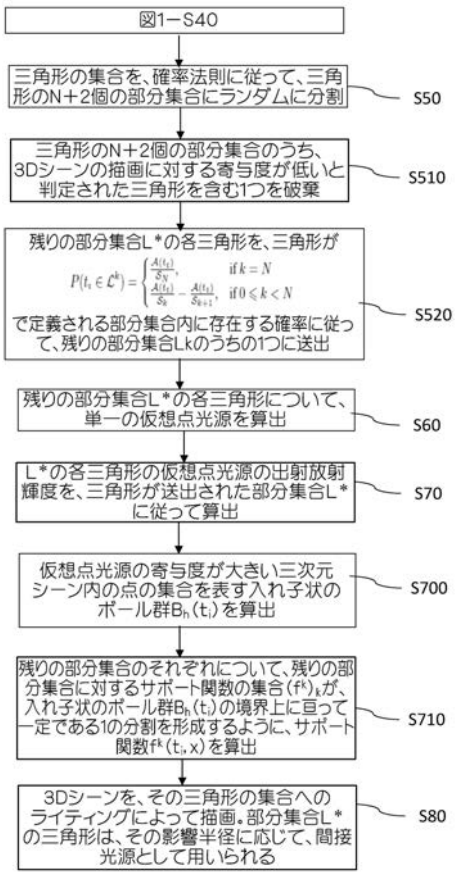
間接照明をシミュレートするためにＶＰＬ寄与度を合計する方法は、前の説明とは関係しない。したがって、シンプルなパイプラインを維持するために、ＳＡＩＴＯ，Ｔ．，ＡＮＤ ＴＡＫＡＨＡＳＨＩ，Ｔ．１９９０．Ｃｏｍｐｒｅｈｅｎｓｉｂｌｅ ｒｅｎｄｅｒｉｎｇ ｏｆ ３－ｄ ｓｈａｐｅｓ．ＡＣＭ ＳＩＧＧＲＡＰＨ Ｃｏｍｐｕｔｅｒ Ｇｒａｐｈｉｃｓ，ｖｏｌ．２４，ＡＣＭ，１９７－２０６で論じられている遅延照明と同様のスプラッティング戦略を用いてもよい。特に、これにより、ジオメトリの間引き、ＶＰＬ生成、ライティングを１つのシェーダプログラムで管理できる。実際には、ジオメトリ・シェーダは、三角形が非標準であるかどうか、および、どれが生成されたＶＰ

Ｌのレベルであるかを判断する以外に、基になるＶＰＬスクリーン空間関数サポートを含むサイズ指定されたポイントプリミティブの入力三角形を変換するのに用いることができる。結果として生じる信号をアンダーサンプリングすることが、良好な最適化である可能性がある。これを行うために、ビューポートは$4^{Ntiling\ level}$のタイルに分割され、ビューポート内の各画素に対し、同じ相対位置に、一意なタイル画素を割り当てる。この技術は、しばしばインターリーブサンプリングと呼ばれる（ＫＥＬＬＥＲ，Ａ．，ＡＮＤ ＨＥＩＤＲＩＣＨ，Ｗ．２００１．Ｉｎｔｅｒｌｅａｖｅｄ ｓａｍｐｌｉｎｇ．Ｓｐｒｉｎｇｅｒ）。次に、スプラッティング時に、ＶＰＬごとにランダムにタイルを選択し、タッチした画素数を$4^{Ntiling\ level}$で除算する。画像は、バッファを非タイル化し、発生したノイズを除去するためにぼかすことによって、再構成される。

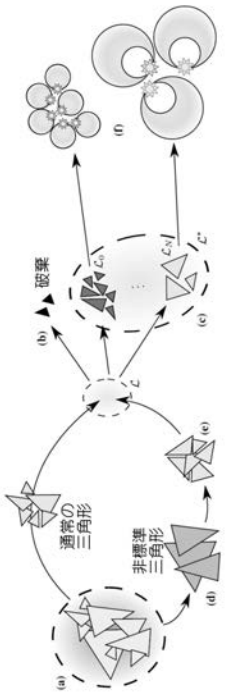
【図１】



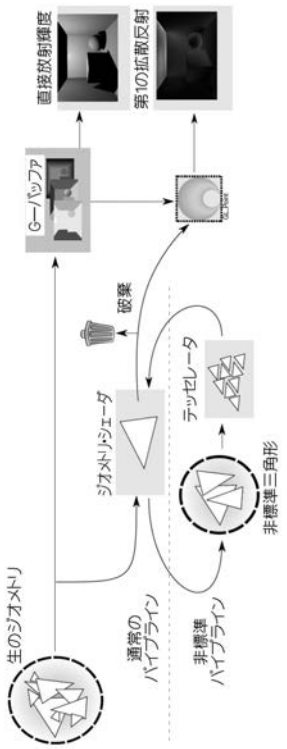
【図２】



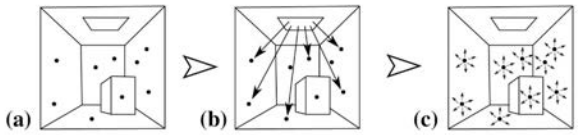
【図３】



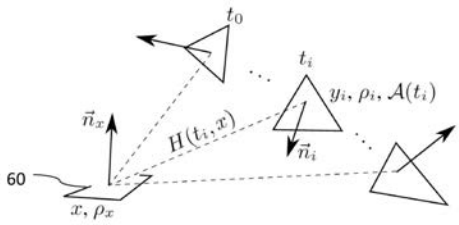
【図４】



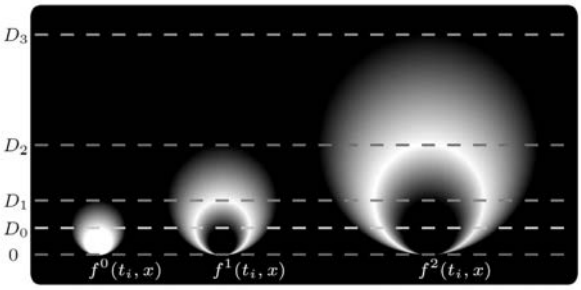
【図５】



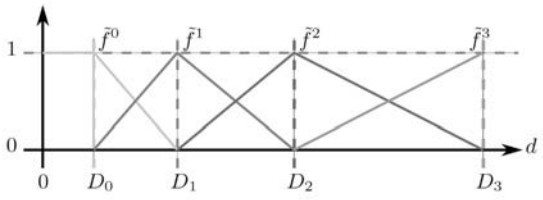
【図６】



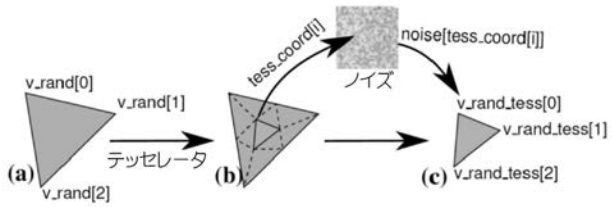
【図７】



【図８】



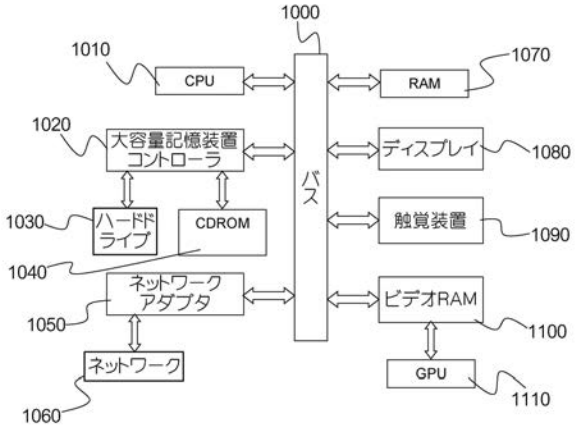
【図９】



【図１０】

【図１１】

【図１２】



フロントページの続き

(72)発明者　グレゴール　デ・ラ・リビエ
フランス国　３３２００　ボルドー　アベニュー　デ・ジーヌ　１６８
(72)発明者　タミー　ブービカー
フランス国　７５０１３　パリ　アベニュー　ド　ショワジー　１７２

Ｆターム(参考) 5B080 AA14　BA00　BA05　CA04　GA08

【外国語明細書】

RENDERING THE GLOBAL ILLUMINATION OF A 3D SCENE

FIELD OF THE INVENTION

The invention relates to the field of computer programs and systems, and more specifically to a method, system and program for rendering the global illumination of a three-dimensional scene in the context of large models such as the complex scenes usually encountered in CAD application scenarios.

BACKGROUND

A number of systems and programs are offered on the market for the design, the engineering and the manufacturing of objects. CAD is an acronym for Computer-Aided Design, e.g. it relates to software solutions for designing an object. CAE is an acronym for Computer-Aided Engineering, e.g. it relates to software solutions for simulating the physical behavior of a future product. CAM is an acronym for Computer-Aided Manufacturing, e.g. it relates to software solutions for defining manufacturing processes and operations. In such computer-aided design systems, the graphical user interface plays an important role as regards the efficiency of the technique. These techniques may be embedded within Product Lifecycle Management (PLM) systems. PLM refers to a business strategy that helps companies to share product data, apply common processes, and leverage corporate knowledge for the development of products from conception to the end of their life, across the concept of extended enterprise. The PLM solutions provided by Dassault Systèmes (under the trademarks CATIA, ENOVIA and DELMIA) provide an Engineering Hub, which organizes product engineering knowledge, a Manufacturing Hub, which manages manufacturing engineering knowledge, and an Enterprise Hub which enables enterprise integrations and connections into both the Engineering and Manufacturing Hubs. All together the system delivers an open object model linking products, processes, resources to enable dynamic, knowledge-based product creation and decision support that drives optimized product definition, manufacturing preparation, production and service.

Light transport simulation is an important component of realistic image synthesis. The body of research work related to this phenomenon is referred as

"global illumination" and, despite its well-known physics laws, remains a challenging problem due to its high computational cost, with even more critical consequences for real-time scenarios. Hidden behind the recursive nature of the rendering equation as discussed in *KAJIYA, J. T. 1986. The rendering equation. In ACM Siggraph Computer Graphics, vol. 20, ACM, 143–150.*, global illumination simulation has been addressed in a large number of approaches, for the high degree of realism it brings to special effects, previsualization, animated movie production and video games.

Currently, two lines of research can be distinguished. The first one is "interactive rendering" which aims at quickly providing a visually convincing approximation of global illumination. The second one is "offline rendering" which targets a solution which is as close as possible to physics.

Among the various strategies that can be used to achieve real-time performance for "interactive rendering", exploiting the low frequency behavior of distant incoming radiance has proven to be effective. For instance, hierarchical radiance caching methods discussed in *WALTER, B., FERNANDEZ, S., ARBREE, A., BALA, K., DONIKIAN, M., AND GREENBERG, D. P. 2005. Lightcuts: a scalable approach to illumination. In ACM Transactions on Graphics (TOG), vol. 24, ACM, 1098–1107* exploit this property by computing a tree over the geometry of the scene, which stores a multiscale radiance function. Assuming indirect lighting being modelled by virtual point lights (VPLs) located on the scene surfaces, every internal node of this tree provides a representative radiance response of its related subtree and is used as a substitute to it as soon as the radiance is evaluated from a distant location. The set of nodes used to evaluate the incoming radiance at a given point (e.g. unprojected image pixel) is called a light cut, since gathering a node induces discarding its children.

A number of high-performance implementations of light cut gathering processes have been proposed, but most of them rely on precomputing and maintaining the tree, which can have a significant memory and computational cost.

Mittring et al. (*MITTRING, M. 2007. Finding next gen: Cryengine 2. In ACM SIGGRAPH 2007 courses, ACM, 97–121*) introduced an ambient occlusion approximation method using the depth-buffer as an economic, random-accessible substitute to the actual (potentially large) geometry of the scene, and parameterizing

the light cache in screen-space. A large variety of other methods exploits screen-space approximations to lower the computational complexity of some lighting effects. However, despite their real-time and dynamic performances, such approaches rely on depth peeling and multiple views rendering to account for the full geometry of the scene i.e., beyond the first depth layer and outside the view frustum, which quickly impacts negatively their native speed.

In contrast to screen-space approaches, solving for indirect illumination in object-space avoids such view-dependent artifacts, at the cost of less GPU-friendly light caches. For instance, Instant Radiosity (IR) methods as discussed in *KELLER, A. 1997. Instant radiosity. In Proceedings of the 24th annual conference on Computer graphics and interactive techniques, ACM Press/Addison-Wesley Publishing Co., 49–56* work with a set of secondary point light sources, called Virtual Point Lights (or VPLs), directly generated on the geometry illuminated by the primary sources. Thus, a VPLs set acts as a discrete representation of the scene's indirect lighting and allows to reduce computations drastically when approximating light bounces. However, scaling up to massive data requires huge amounts of VPLs. This is challenging as both generation and shading costs of so many VPLs is prohibitive in dynamic scenes.

Finally, the Deep Screen Space (DSS) approach discussed in *NALBACH, O., RITSCHEL, T., AND SEIDEL, H.-P. 2014. Deep screen space. In Proceedings of the 18th meeting of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games, ACM, 79–86. 2014* proposes to exploit the advantages of both screen-space and object-space radiance caching. The same way as object-space strategies, this method generates on-surface VPLs, even on occluded geometry that may still impact the image; and similarly to screen-space approaches, it benefits from a native GPU support, with the tessellator unit – instead of the rasterizer – being used as a surface sampler to generate the VPLs. Still, although DSS can successfully be used for rendering small to medium size scenes, it cannot cope with larger ones, where the real-time constraints imposes decimating geometry rather than refining it.

Within this context, there is still a need for an improved method rendering the global illumination of a three-dimensional scene.

SUMMARY OF THE INVENTION

It is therefore provided a computer-implemented method for rendering the global illumination of a three-dimensional scene. The method comprises:

- providing a three-dimensional scene that comprises of a set $\mathcal{L}$ of triangles and one or more direct light sources;
- determining that each triangle t_i of the set has an area that is below a threshold S_o;
- assigning to each triangle of the set a radius of influence using a probability law;
- obtaining a subset of triangles $\mathcal{L}^*$ by filtering out the triangles according to their radius of influence;
- rendering the three-dimensional scene by lighting its set of triangles, the triangle of the subset of triangles $\mathcal{L}^*$ being used as indirect light sources according to their radius of influence.

The method may comprise one or more of the following:

- determining that each triangle of the set has an area that is below the threshold comprises partitioning the set of triangles $\mathcal{L}$ into a first set of triangles, wherein each triangle of the first set has an area that is below the threshold; a second set of triangles, wherein each triangle of the second set has an area that is above the threshold; and tessellating each triangle of the second set into a third set of triangles, wherein each triangle of the third set has an area that is below the threshold as a result of the tessellation; and wherein assigning to each triangle of the set a radius of influence using a probability law comprises assigning to each triangle of the first and the third sets a radius of influence using a probability law; and obtaining a subset of triangles comprises obtaining a subset of triangles by filtering out the triangles of the first and third sets according to their radius of influence;
- comprising, after determining that each triangle of the set has an area that is below a threshold: randomly partitioning the set of triangles into $N + 2$ subsets of triangles according to the probability law; discarding one of the $N + 2$ subsets of triangles that comprises triangles that are determined as having a low contribution to the rendering of the 3D scene, the remaining N+1 subsets comprising non-discarded triangles;

- the remaining subsets $\mathcal{L}^*$of the $N + 2$ subsets of triangles comprise triangles t_i of the set such that

$$\forall t_i \in \mathcal{L},\ P(t_i \in \mathcal{L}^*) = \frac{\mathcal{A}(t_i)}{\mathcal{S}_0},$$

wherein $A(t_i)$ is the surface area of the triangle t_i and S_o is the threshold;

- each triangle of the remaining subsets $\mathcal{L}^*$is dispatched (S520) in one of the remaining subsets $\mathcal{L}^k$ according to a probability for a triangle to lie in a subset $\mathcal{L}^k$ defined by:

$$P(t_i \in \mathcal{L}^k) = \begin{cases} \frac{\mathcal{A}(t_i)}{\mathcal{S}_N}, & \text{if } k = N \\ \frac{\mathcal{A}(t_i)}{\mathcal{S}_k} - \frac{\mathcal{A}(t_i)}{\mathcal{S}_{k+1}}, & \text{if } 0 \leqslant k < N \end{cases}$$

wherein the value of S_k increases for when k increases;

- S_k is set such that $S_k = S_o\mu^k$, where $\mu > 1$ and the value of μ is a real number comprised between $[1; 5]$;

- S_k is set such that $S_k = S_o\mu^k$, wherethe value of μ is defined by $\mu > \sqrt[N]{4\frac{\mathcal{R}^2_{scene}}{S_o}}$ with $\mathcal{R}_{scene}$that is the radius of the three-dimensional scene;

- after that each triangle of the remaining subsets $\mathcal{L}^*$has been dispatched: computing a single virtual point light for each triangle of the remaining subsets $\mathcal{L}^*$; and wherein assigning to each triangle of the set a radius of influence using a probability law further comprises computing an outgoing radiance for the virtual point light of each triangle of the remaining subsets $\mathcal{L}^*$, the computation being performed according to the subset in which the triangle has been dispatched;

- computing an outgoing radiance for the virtual point light of each triangle comprises computing a family of nested balls $\mathcal{B}_h(t_i)$ representing the set of points in the three-dimensional scene for which the contribution of the virtual point lights is significant;

- computing, for each remaining subset of the $N + 2$ subsets, a support function $f^k(t_i, x)$ such that the set of support functions $(f^k)_k$ over the remaining subsets form a partition of unity that is constant over a frontier of a $\mathcal{B}_h(t_i)$ of the family of nested balls;

- a base function of the set of support functions $(f^k)_k$is a B-spline;

- $\mathcal{B}_h(t_i)$ is defined by $\mathcal{B}_h(t_i) = \{x \epsilon \mathbb{R}^3 \; s.t. \frac{||x-y_i||}{\langle \overrightarrow{n_x}, \; \vec{x}y_i \rangle^+} \leq D(h) = \frac{1}{\pi}\sqrt{\frac{\rho_x \rho_i E(t_i)}{h}}\}$, wherein x is a point of the three-dimensional scene with normal $\overrightarrow{n_x}$, y_i is the point on which the virtual point light is centered on the triangle t_i, ρ_x is albedo at point x, ρ_i is albedo at point y_i, $E(t_i)$ is a direct irradiance falling to the triangle t_i;

- at the rendering step, the set of support functions $(f^k)_k$ is used for carrying out splatting operations when rendering a sphere of a $\mathcal{B}_h(t_i)$ of a family of nested balls;

- the threshold S_o is defined by $S_o = 4\pi \frac{D_{near}^2}{N_{Avg}}$, wherein N_{Avg} is an average number of virtual point lights, D_{near} is a minimal distance between pixels of the 3D scene and VPLs;

- before determining that each triangle t_i of the set has an area that is below a threshold S_o: generating (S20) a single random integer for each vertex of triangle t_i of the set;

- wherein computing a single random integer for each vertex of triangle further comprises for each sub-vertex of a triangle of the third set, determining a barycentric coordinate of the sub-vertex; using the barycentric coordinate to fetch (S210) a random value in a noise texture; using the fetched random value to generate (S220) the single random integer of the said each sub-vertex of a triangle of the third set.

It is further provided a computer program comprising instructions for performing the method. It is further provided a graphic library computer program comprising instructions for performing the method.

It is further provided a computer readable storage medium having recorded thereon the graphic library computer program.

It is further provided a system comprising a processor coupled to a memory and a graphical card with a graphic processing unit, the memory having recorded thereon the computer program.

The graphical card of the system may comprise a geometry shader unit for determining that each triangle t_i of the set has an area that is below a threshold S_o; and a tessellation unit for tessellating triangles t_i of the set; wherein the computer

program recorded on the memory is adapted to configure the tessellation unit for performing the generation of the single random integers.

BRIEF DESCRIPTION OF THE DRAWINGS

Embodiments of the invention will now be described, by way of non-limiting example, and in reference to the accompanying drawings, where:

- Figures 1 and 2 show a flowchart of an example of the method;
- Figure 3 shows an example of the method;
- Figure 4 shows an example of indirect illumination shader pipelines of the invention;
- Figure 5 shows an example of indirect illumination pipeline of the invention;
- Figure 6 shows an example of the illumination of a virtual point light at a point x;
- Figure 7 shows an example of visualization of support functions on a planar section, for values ranging from 0 (black) to 1 (white);
- Figure 8 shows an example of one dimensional partition of unity;
- Figure 9 shows an example of generation of per-triangle uniform random values;
- Figures 10 and 11 are screenshots showing examples of illuminations of a same scene without and with the divergent pipeline of the invention;
- Figure 12 shows an example of a system for performing the method.

DETAILED DESCRIPTION OF THE INVENTION

With reference to the flowcharts of Figures 1 and 2, it is proposed a computer-implemented method for rendering the global illumination of a three-dimensional scene. The method comprises providing a three-dimensional scene that comprises of a set $\mathcal{L}$ of triangles and one or more direct light sources. Then, one determines that each triangle t_i of the set has an area that is below a threshold S_o. Next, a radius of influence is assigned to each triangle of the set using a probability law. Then, a subset of triangles $\mathcal{L}^*$ is obtained by filtering out the triangles according to their radius of influence. Next, the three-dimensional scene is rendered by lighting its set of triangles, the triangle of the subset of triangles $\mathcal{L}^*$ being used as indirect light sources according to their radius of influence.

Such a method improves the rendering in real-time of the global illumination of a three-dimensional scene. Indeed, a multiscale representation of lighting is generated – VPLs are generated based on a stochastic decimation process of the input triangle set - and forwardly (i.e. exploring light space from sources to sensor) – compute the bounded influence region of each of them as if they were selected in a cut. The present exploits both the geometry shader and the tessellator unit of a graphic card to build, on a per frame basis, the set of virtual point lights (VPLs) that approximate indirect lighting. The present invention proposes a diffuse graphic pipeline for illumination which can both refine and simplify the set of geometry driven VPLs in a two-pass strategy. Exploiting both the tessellator unit and the geometry shader to adjust the resolution of an object space radiance cache in the context of scenes with a massive number of triangles. The present invention does not imply any complex preprocessing nor requires carrying complex data structures over frames. Real-time performance is reached by using a multiscale representation of the light field, and is fully dynamic and does not resort to any tree structure, nor imposes to maintain any data structure among frames. It is compatible with fully dynamic scenes, including light, view point and geometry animations. Last, but not the least, in terms of integration, the present invention naturally fits standard modern graphics pipelines and does not make any strong assumption on the complementary rendering techniques employed by the host application.

The method is computer-implemented. This means that the steps (or substantially all the steps) of the method are executed by at least one computer, or any system alike. Thus, steps of the method are performed by the computer, possibly fully automatically, or, semi-automatically. In examples, the triggering of at least some of the steps of the method may be performed through user-computer interaction. The level of user-computer interaction required may depend on the level of automatism foreseen and put in balance with the need to implement user's wishes. In examples, this level may be user-defined and/or pre-defined.

A typical example of computer-implementation of the method is to perform the method with a system adapted for this purpose. The system may comprise a processor coupled to a memory and a graphical card with a graphic processing unit,

the memory having recorded thereon a computer program comprising instructions for performing the method. The computer program can be a graphic library computer program. The memory may also store a database. The memory is any hardware adapted for such storage, possibly comprising several physical distinct parts (e.g. one for the program, and possibly one for the database). The database may store the 3D scene to be rendered, e.g. the database stores the set of triangles of the scene.

Figure 12 shows an example of the system, wherein the system is a client computer system, *e.g.* a workstation of a user.

The client computer of the example comprises a central processing unit (CPU) 1010 connected to an internal communication BUS 1000, a random access memory (RAM) 1070 also connected to the BUS. The client computer is further provided with a graphical processing unit (GPU) 1110 which is associated with a video random access memory 1100 connected to the BUS. Video RAM 1100 is also known in the art as frame buffer. A mass storage device controller 1020 manages accesses to a mass memory device, such as hard drive 1030. Mass memory devices suitable for tangibly embodying computer program instructions and data include all forms of nonvolatile memory, including by way of example semiconductor memory devices, such as EPROM, EEPROM, and flash memory devices; magnetic disks such as internal hard disks and removable disks; magneto-optical disks; and CD-ROM disks 1040. Any of the foregoing may be supplemented by, or incorporated in, specially designed ASICs (application-specific integrated circuits). A network adapter 1050 manages accesses to a network 1060. The client computer may also include a haptic device 1090 such as cursor control device, a keyboard or the like. A cursor control device is used in the client computer to permit the user to selectively position a cursor at any desired location on display 1080. In addition, the cursor control device allows the user to select various commands, and input control signals. The cursor control device includes a number of signal generation devices for input control signals to system. Typically, a cursor control device may be a mouse, the button of the mouse being used to generate the signals. Alternatively or additionally, the client computer system may comprise a sensitive pad, and/or a sensitive screen.

The computer program may comprise instructions executable by a computer, the instructions comprising means for causing the above system to perform the method. The program may be recordable on any data storage medium, including the memory of the system. The program may for example be implemented in digital electronic circuitry, or in computer hardware, firmware, software, or in combinations of them. The program may be implemented as an apparatus, for example a product tangibly embodied in a machine-readable storage device for execution by a programmable processor. Method steps may be performed by a programmable processor executing a program of instructions to perform functions of the method by operating on input data and generating output. The processor may thus be programmable and coupled to receive data and instructions from, and to transmit data and instructions to, a data storage system, at least one input device, and at least one output device. The application program may be implemented in a high-level procedural or object-oriented programming language, or in assembly or machine language if desired. In any case, the language may be a compiled or interpreted language. The program may be a full installation program or an update program. Application of the program on the system results in any case in instructions for performing the method.

As it will discussed in greater details below, the present invention uses Virtual Point Lights (VPLs), i.e. a set of points for which a certain lighting is cached. The cached lighting may be direct lighting. VPLs are used to approximate indirect light bounces. Figure 5 shows an example of VPL based pipeline, as known in the art. A VPL pipeline is typically composed of three main stages that are (a) VPLs generation, (b) indirect light caching and (c) lighting with VPLs.

The present invention notably focusses on (a) VPLs generation and (c) their usage during lighting. Figure 1 shows an example of the generation of triangles for which VPLs are generated:

- at preprocessing time, the computing of a single per-vertex random integer which will be used at rendering time to generate per triangle random numbers consistently, even under dynamic geometry transformations;

- an example of the creation of a subset of triangles from the raw geometries is discussed. Notably the use of a regular pipeline and divergent pipeline is presented; large triangles – that are called "divergent" in the present invention – are not well sampled by the aforementioned strategy and may introduce important lighting artifacts in the final image; consequently, divergent triangle are routed through a tessellation unit, where they are subdivided until reaching the proper resolution;

- an example of the discard of triangles of the raw data, e.g. in order to greatly reduce the number of managed triangles while optimizing the amount of sampling information, a special set – mainly composed of small triangles – is completely discarded.

Figure 2 shows an example of the generation of the VPLs, wherein the full set of triangles generated from the raw data (Figure 1) is randomly partitioned according to a probabilistic law. For each triangle, a VPL is generated and its outgoing radiance is computed based solely on its related partition ($\mathcal{L}^0, \ldots \mathcal{L}^N$). Finally, indirect illumination is splatted in a typical deferred shading process, with the splatted function supports depending on the VPL partition, reserving powerful VPLs to carry on distant lighting using crescent-shaped support.

Referring back to Figure 1, a first aspect of the present invention is now discussed, which is the generation of triangles from raw data for which VPLs are generated. At step **S10**, a 3D scene is provided. The 3D scene comprises a set $\mathcal{L}$ of triangles and one or more direct light sources. Providing means that the raw data (the set $\mathcal{L}$ of triangles of the 3D scene) can be accessed by the system and/or the program implementing the method of the invention. Typically, the 3D scene can be loaded in a Render Engine that is a framework that generates images to be displayed upon application's request, as known in the art. For instance, a CAD application of CAD system provides to the Render Engine a 3D scene of 3D modeled objects as input and the Render Engine draws the 3D scene to the screen with the use of one or more Graphic Cards of the CAD system. The triangles of the 3D scene form a mesh, e.g. modeled objects of the 3D scene are represented by interconnected triangles, as known in the art.

Then, at step **S20**, a single random integer is generated for each vertex of each triangle of the set $\mathcal{L}$. As a result, a per-vertex uint32 attribute – v_rand is added. This attribute is initialized when loading the mesh with random values uniformly chosen between, but is not limited to this range, 0 and 2^{32}, or even between 0 and 2^{64}. It is to be understood that this step can be performed as known in the art.

The steps S10 and S20 are noted (a) on Figure 3.

Then, at step **S30**, it is determined that each triangle t_i of the set has an area that is below a threshold S_o. Depending on the result of the determination, the raw data is split (or not) into regular and divergent triangles of the set, wherein divergent triangles are triangles with an area greater than the threshold S_o that are subdivided such that every new triangle may be considered as regular as illustrated by (d) and (e) on Figure 3; Regular triangles are triangles with an area smaller or equal to the threshold S_o. Steps S310-S312 discuss the regular pipeline of the regular triangles, and steps S320-S322 discussed the divergent pipeline.

Regular triangles are distinguished from divergent ones, i.e. the set of triangles t_i with surface area $\mathcal{A}(t_i)$ greater that a certain threshold S_o. This threshold can be set such that (equation 1)

$$S_o = 4\pi \frac{D_{near}^2}{N_{Avg}}$$

which is a heuristic aiming at approximately lighting pixels with N_{Avg}VPLs at least D_{near}far from them. D_{near} is a variable that provides a minimal distance between the pixels and the VPLs. N_{Avg} is a number of VPLs on the 3D scene. Let R_{scene}be the scene radius, set D_{near}can be set $D_{near} = 0.2 \times R_{scene}$ and N_{Avg} is preferably set between 64 and 1024 depending on the desired quality/speed tradeoff. It is to be understood that the range of value of N_{Avg} may vary and that the number of VPLs can be higher depending on the computing resources of the system performing the rendering. The diameter of the scene is set $Diam_{scene}$= $2\times R_{scene}$ and the $Diam_{scene}$is the maximal distance between two points of the scene.

Hence, at step **S300**, the set of triangles t_i is partitioned in to a first set of triangles (**S310**) with an area smaller or equal to S_o, and a second set of triangles (**S320**) wherein each triangle of the second set has an area that is above S_o. It is to

be understood that the first set might comprise the triangles smaller to S_o and the second set might comprise the triangles equal or aboveS_o.

At step **S312**, the triangles of the first set a stored, e.g. on a cache memory.

At step **S322**, the triangles of the second set are tessellated into a third set of triangles. The tessellation is performed as known in the art. The tessellated triangles have an area that is smaller or equal to S_o: each triangle of the third set has an area that is below the threshold as a result of the tessellation.

Now, the raw data comprises two set of triangles. Triangles of both sets have an area that is smaller or equal to S_o: the difference between both sets is that each vertex of each triangle of the first set is associated with a single random integer, which is not the case for the triangles of the second set because the tessellation creates new triangles for which one or more vertices do not have a single random integer previously generated. In the example of Figure 1, the single random integer has been generated and associated to the raw data. It is to be understood that single random integer could be generated and associated at a later stage, for instance while the first and second set are being determined ; or even after the partitioning of the raw data have been performed. For example, once the first set has been stored, the single random integers might be generated for the vertices of the triangles of the first set.

Steps S200-S220 provide an example for generating a single random integer for each vertex of each triangle of the second set. Preferably, these steps are carried out while tessellating the triangles of the second set: for each newly tessellated triangle of the second set (S320), the steps S200 to S220 are performed. Alternatively, they can be performed after the third set have been build. Here, the steps S200-S220 are discussed in the case the generation is performed each time a triangle of the second set is tessellated.

At step **S200**, one determines each sub-vertex of a triangle of the third set. As illustrated on Figure 9-(a), each vertex of the triangle of the second set comprises a uint32 attribute – v_rand. In Figure 9-(b), the sub-vertices 90a, 90b, 90c of the triangle 90 are obtained, being understood that the triangle 90 is one of the new ten triangles obtained from the triangle of Figure 9-(a).

Then, at step **S210**, one identifies the barycentric coordinates tess_coord[i] of the said each sub-vertex of the triangle of the third set. The barycentric coordinates are computed during the tessellation of the triangle of the second set from which the said each sub-vertex of a triangle of the third set was obtained. A random value is then fetched in a noise texture by using the barycentric coordinates tess_coord[i]. More precisely, for each tess_coord[i], a random value is fetched. The fetching is performed as known in the art: a texture can be contemplated as a bi-dimensional table where each cell of the table stores a value, for instance the values are 32bits integers. When a random value is fetched, one of the cell is read, the cell been reached in the table by use of the coordinate tess_coord[i]. The noise texture is in general precomputed, e.g. at the time the rendering of a 3D scene is requested. The noise texture is a table of a given size, for instance 1024x1024.

Next, at step **S220**, the fetched random value is used to generate the single random integer of the said each sub-vertex of a triangle of the third set. As illustrated on Figure 9-(c), each vertex of the triangle 90 is associated with a random value v_rand_tess[1] generated from the barycentric coordinate tess_coord[i] of this vertex on the tessellated triangle.

Some triangles of the third set are already associated with random values, the one generated at step S20. These random values are kept and only the sub-vertices without random value are concerned by steps S200 to S220. Thus one determines sub-vertices v_i of a triangle of the third set without a single random integer. This allows preserving computing resources.

Now, the first and third sets have triangles having an area that is smaller or equal to $S_o$and with vertices associated with a single random value. The first and third sets can be merged (S40) as the divergent pipeline transformed the divergent triangles into regular triangles.

Referring now to Figure 2, it is discussed the generation of the VPLs in accordance with a probability law and the way the merged first and third sets are randomly partitioned.

At step **S50**, the set of triangles (S40) is partitioned into N+2 subsets of triangles according to a probability law, and one of these N+2 subsets is discarded (S510), that is, the triangles are definitively removed.

In practice, the decimation of the triangles at step S510 can be performed before the partitioning so that less computing resources are required for creating the partitions as less triangles have to be analyzed when partitioning.

The triangle decimations aims at removing small triangles of the 3D scene as they contribute poorly to the final rendering for diffuse indirect lighting. A straight forward solution would be to remove small triangles under an area threshold. However, groups of small triangles may collectively have an important impact in the light transport. A stochastic decimation approach is used in which the contribution of heavily tessellated geometry is retained by computing, for each triangle of the first and third set, a uniform random value u_{ti} lying between 0 and 1. The rule is the following: if $\mathcal{A}(t_i) > u_{ti}S_o$, then the triangle is kept (that is, not discarded), and if $\mathcal{A}(t_i) \leq u_{ti}S_o$ then the triangle is added to the set of triangles to be discarded (being understood that the triangle can be immediately discarded).

If one takes the assumption that every triangle can be assumed to be regular, the probability for a triangle t_i to be kept boils down to (equation 2):

$$\forall t_i \in \mathcal{L},\ \ P(t_i \in \mathcal{L}^*) = \frac{\mathcal{A}(t_i)}{S_0}$$

where $\mathcal{L}^*$denotes the triangles t_i of the first and third sets (S40) that are kept, and $\mathcal{L}$ denotes the triangles of the first and third sets. Intuitively, this means that the smaller a triangle is, the greater its chance to be discarded becomes. At the same time, this partitioning translates into a uniform distribution of samples over the entire scene surface, such that the kept triangle count expectation is $\mathbb{E}(N_{sample}) = \frac{\mathcal{A}_{scene}}{S_o}$ with $\mathcal{A}_{scene}$ that is the total scene area.

Once the triangles have been discarded, or at least associated with a partition to be discarded, the remaining triangles (that is, the triangles that are kept) are dispatched (step S520) in one ($\mathcal{L}^0$, ... $\mathcal{L}^N$) of the N+1 remaining subsets $\mathcal{L}^k$ according to a probability for a triangle to lie in a subset $\mathcal{L}^k$. The dispatch is such that that $\mathcal{L}^N$ contains a few triangles and $\mathcal{L}^k$ is more and more populated when k gets closer to 0.

Doing so, a multiscale partitioning of representative scene triangles emerges from the decimation strategy (step S510), with the probability for a triangle to lie in the subset $\mathcal{L}^k$ being (equation 3):

$$P(t_i \in \mathcal{L}^k) = \begin{cases} \frac{\mathcal{A}(t_i)}{S_N}, & \text{if } k = N \\ \frac{\mathcal{A}(t_i)}{S_k} - \frac{\mathcal{A}(t_i)}{S_{k+1}}, & \text{if } 0 \leqslant k < N \end{cases}$$

and (equation 4)

$$S_k = S_o \mu^k$$

where $\mu > 1$ is a user defined real number. With $S_k = S_o \mu^k$, the number of VPLs per subset $\mathcal{L}^k$ evolves exponentially, which mimic a behavior of some existing techniques of the prior art, e.g. those based on tree structures.

The value of μ depends on the scene and the number of partitions. The value μ is typically comprised between 1,4 and 5. A requirement on acceptable values of μ will be discussed later in a discussion regarding an example of implementation of the method. Indeed, $S_k = S_o \mu^k$ is an implementation choice and may be understood as the average surface of triangles lying in the partition (or subset) $\mathcal{L}^k$. In practice, $S_k$is chosen so that it value increases when k increases. With the aforementioned formulation (Equation 4), the partitioning induces subsets with a size expectation that decreases geometrically, mimicking the traditional hierarchical representations used with light fields. However, no kind of explicit hierarchy is maintained, generated, or managed because triangles are affected to a certain subset independently from the choice made for any other. In particular, this means that a given triangle, located in a given subset, does not capture any coarse-grained information carried by finer triangles.

The following algorithm 1 shows an example of the generation of a partition such as defined from equation 3, with the probability law u_{ti} kept unchanged.

```
1: procedure COMPUTELEVEL(u_ti ; ti)
2:   for k ← N ... 1 do
3:       if u_ti < A(t_i)/S_k then
4:           return k
5:       end if
6:   end for
7:   DISCARDTRIANGLE( )fthreshol
```

To evaluate algorithm 1 for each triangle, a pseudo-random value is used for the variable u_{ti}. In an example, the generation of this value relies on the additional per-vertex uint32 attribute – v rand. As discussed, this attribute is initialized when loading the mesh (the raw data) with random values uniformly chosen for example between 0 and 2^{32}. In the regular pipeline, at geometry shading stage, u_{ti} is computed as a xor between the random attributes of the three vertices of a triangle ti. The choice of the xor operator allows avoiding correlation among two or three triangles. Moreover, new random values can be generated at any time by using u_rand that is a global uniform random value that may be updated at most once per frame at the rendering time. Being xor'd with the original per-vertex values, it allows to get whole new random distributions over the mesh. This is apparent in the algorithm 2 below.

For the divergent pipeline (for triangle that are not regular), the construction of u_{ti} is quite more subtle. Indeed, this value remains unaltered, even under camera motion or mesh deformation, i.e. as long as the mesh topology remains the same. To preserve these properties, one performs the tessellation in the model space and exploit the fact that the tessellation pattern only depends on the input triangle shape. Indeed, the tessellation parameters are only determined by the original triangle area, as explained in the discussion regarding steps S200-S220. Thus, to generate u_{ti} for a triangle sprung from the subdivision, a xor is computed between the three v_rand_tess, the three base triangle v_rand and finally the global uniform random value u rand, as show in the algorithm 2:

```
1: uniform uint32 u rand
2: uniform texture2D noise
3: function REGULARRAND(ivec3 v rand)
4:     return u rand xor v rand.x xor v rand.y xor v rand.z
5: end function
6: function DIVERGENTRAND(ivec3 v rand, vec2 tess coord[3])
7:     ivec3 v tess rand
8:     v rand tess.x = TEXTURE(noise, tess coord[0])
9:     v rand tess.y = TEXTURE(noise, tess coord[1])
10:    v rand tess.z = TEXTURE(noise, tess coord[2])
11:    return REGULARRAND(v rand xor v tess rand)
12: end function
```

Figure 3-(b) and (c) shows the generation of the partitions, that is, the surviving triangles (those that remain after the discard of the "small" triangle (b)) are classified among subsets $\mathcal{L}^0, \ldots \mathcal{L}^k, \ldots \mathcal{L}^N$ according to a probability law $P(t_i)\epsilon\mathcal{L}^k$ of equation 3 with a pseudo-random value used for the variable u_{ti}.

Back to Figure 2, at step S60, a single virtual point light is computed for each triangle of the remaining subsets $\mathcal{L}^*$. Preferably, to properly cope with hardware restrictions, no more than one VPL per triangle is computed; being understood some triangles might have more than one VPL, e.g. the decision could be performed according to their $\mathcal{A}(t_i)$. The computation of the VPL for each remaining triangle is performed as known in the art.

The computing of VPLs actually involves the computing of an outgoing radiance for each VPL in order to perform a lighting with the VPLs. As it will apparent with the mathematical discussion set forth below, the computation of the outgoing radiance (step S70) is performed according to the subset in which the triangle has been dispatched. Hence, a radius of influence (that is, an outgoing radiance) is assigned to each triangle of the first and third sets using a probability law, the one of equation 3 used for obtaining the subsets $\mathcal{L}^0, \ldots \mathcal{L}^k, \ldots \mathcal{L}^N$. Therefore, the probability law can be contemplated as a distribution of radius of influence that is performed so that a decreasing density of radius of influence is obtained: indeed, the partitioning induces subsets with a size expectation that decreases geometrically, mimicking the traditional hierarchical representations used with light fields. One understands that the discarding of the triangles with low contribution to the rendering of the 3D scene amounts to obtaining a subset of the initial set (the raw data) of triangles by filtering out the triangles according to their radius of influence.

Step S700 to 710 show an example for computing outgoing radiance for the virtual point light of each triangle of $\mathcal{L}^*$according to the subset $\mathcal{L}^*$ in which the triangle has been dispatched. A family of nested nested balls $\mathcal{B}_h(t_i)$ is computed. $\mathcal{B}_h(t_i)$ represent the set of points in the three-dimensional scene for which the contribution of the virtual point lights is significant (S700). A support function $f^k(t_i,$ x) is computed for each remaining N+1 subset, at step S710. The set of support functions $(f^k)_k$over the remaining subsets form a partition of unity that is constant

over a frontier of a $\mathcal{B}_h(t_i)$of a family of nested balls. For the sake of explanation, the model of outgoing radiance used in this example is now discussed, and the computations of steps S700 and S710 will be performed accordingly. Figure 3 (f) shows examples of radius of influence with different sizes; the outgoing radiance computed for the virtual point light of a triangle depends on the subset the triangle belongs to.

The indirect outgoing radiance $L(x, \overrightarrow{n_x})$ of a point x with normal n_x is defined by the equation 5 as

$$L(x, \overrightarrow{n_x}) = \sum_{t_i \epsilon \mathcal{L}} H(t_i, x, \overrightarrow{n_x}) \mathcal{A}(t_i)$$

where $\mathcal{L}$ represents the set of all triangles in the scene (that is the raw data) and $H(t_i, x, \overrightarrow{n_x})$ stands for the incoming radiance transfer function starting from t_i toward the receiver x oriented by $\overrightarrow{n_x}$. Figure 6 shows a pixel 60 (the receiver) with a point x oriented by $\overrightarrow{n_x}$ and ρ_x is albedo at point x. For a diffuse receiver with albedo ρ_x, this transfer function is defined as (equation 6)

$$H(t_i, x, \vec{n}_x) = L(t_i, y_i \bar{} x) \frac{\rho_x}{\pi} \frac{\langle \vec{n_x}, x \bar{y}_i \rangle^+ \langle \vec{n_i}, y_i \bar{} x \rangle^+}{d_i^2},$$

where $\bar{u} = \frac{\vec{u}}{\|\vec{u}\|}$, $\langle \vec{u}, \vec{v} \rangle^+ = \max(0, \langle \vec{u}, \vec{v} \rangle)$, $L(t_i, y_i \bar{} x)$

is the radiance leaving the VPL centered at $y_i \epsilon t_i$ toward the direction $\bar{x} y_i$ and $d_i = max(\epsilon, \| \overrightarrow{x y_i} \|)$ is the distance between $y_i$and x clamped to a user parameter ϵ to avoid singularities. Typically, ϵ has a value included in the range [10^{-3}; 10^{-6}].

The first diffuse bounce of light can be modeled with the following VPLs outgoing radiance expression (equation 7):

$$L(x, \overrightarrow{n_x}) = \frac{E(t_i) \rho_i}{\pi} \langle \overrightarrow{n_i}, \overrightarrow{y_i x} \rangle^+$$

where $E(t_i)$ is the direct irradiance falling to the triangle t_i. Note that because of the term $\langle \overrightarrow{n_i}, \overrightarrow{y_i x} \rangle^+$, reflectors cannot be considered as perfectly lambertian anymore. Although light is therefore preferably reflected in the direction of the geometric normal, experiments show that no noticeable changes appear, while this greatly alleviates upcoming computations as it will be discussed below.

In order to cope with hardware limitation, the computation of $L(x,\overrightarrow{n_x})$ may be approximated by summing the contribution of a subset of VPLs, that is, by summing the contributions of the VPL of triangles of a subset $\mathcal{L}^k$. Thus, an estimator noted $K(x,\overrightarrow{n_x})$ of $L(x,\overrightarrow{n_x})$ can be defined (equation 8)

$$K(x,\overrightarrow{n_x}) = \sum_{k=0}^{N} \sum_{t_i \epsilon \mathcal{L}} H(t_i,\ x,\overrightarrow{n_x})\ F^k(t_i,x)$$

where $F^k(t_i,x)$ is an unknown function of the position x, the emitter t_i and the index k. Its equation is derived below.

By computing the expectation of $K(x,\overrightarrow{n_x})$ over the set of every possible partition $(\mathcal{L}^0, \ldots \mathcal{L}^k, \ldots\ \mathcal{L}^N)$, one obtains the following equation 9

$$\begin{aligned}\mathbb{E}\left[K(x,\vec{n}_x)\right] &= \mathbb{E}\left[\sum_{k=0}^{N} \sum_{t_i \in \mathcal{L}^k} H(t_i^k, x, \vec{n}_x) F^k(t_i^k, x)\right] \\ &= \sum_{t_i \in \mathcal{L}} H(t_i, x, \vec{n}_x) \mathbb{E}\left[\sum_{k=0}^{N} F^k(t_i, x) \mathbb{1}_{[t_i \in \mathcal{L}^k]}\right] \\ &= \sum_{t_i \in \mathcal{L}} H(t_i, x, \vec{n}_x) \sum_{k=0}^{N} F^k(t_i, x) P(t_i \in \mathcal{L}^k),\end{aligned}$$

where $\mathbb{1}_{[t_i \in \mathcal{L}^k]}$ is the indicator function, that equals to 1 if $t_i \epsilon \mathcal{L}^k$ and 0 otherwise.

If one wants $K(x,\overrightarrow{n_x})$ to represent an unbiased estimator of the incoming radiance $L(x,\overrightarrow{n_x})$, it is necessary to verify the following functional equation on the function F^k (equation 10)

$$\forall x,\ \textstyle\sum_k F^k(t_i,x)\ P(t_i \epsilon \mathcal{L}^k) = \mathcal{A}(t_i).$$

According to the VPLs partitioning strategy as established in the equation 3, F^k is defined as (equation 11)

$$F^k(t_i,x) = \begin{cases} \mathcal{S}_0 \mu^N f^N(t_i,x), & \text{if } k = N \\ \mathcal{S}_0 \frac{\mu^{k+1}}{\mu - 1} f^k(t_i,x), & \text{if } 0 \leqslant k < N \end{cases},$$

Equation 11 allows to rewrite the unbiased condition of equation 10 as a partition of unity problem. The partition of unity is a set of functions from a set X to the unit

interval [0,1] such that for every point, x in X, the sum of all the function values at x is 1. This leads to seeking for a set of functions $(f^k)_k$ such that (equation 12)

$$\forall x, \sum_k f^k(t_i, x) = 1.$$

Thus, the lighting with VPLs has been approximated using the partitioning of the triangles into subsets $(\mathcal{L}^0, \ldots \mathcal{L}^k, \ldots \mathcal{L}^N)$, that is, it has been approximated by use of the probability law used for dispatching the triangles into the N+1 subsets. This approximation also leads to find out a set of functions $(f^k)_k$, each function of the set being a function that supports the incoming radiance of a VPL of a subset. The support of the function is the set of points where the function is not zero-valued. Thus, for each subset $(\mathcal{L}^0, \ldots \mathcal{L}^k, \ldots \mathcal{L}^N)$, a support function is computed that depends on the support it belongs to. For instance, the support of the function $f^k$is $\mathcal{B}_{D_{k+1}}$, as shown in equation 17-18. This means that the support of $f^k$is one of the function $\mathcal{B}_h$for a particular h such that $h = D_{k+1}$.

Inspired from PBGI tree cuts strategies discussed in *CHRISTENSEN, P. 2008. Point-based approximate color bleeding. Pixar Technical Notes 2, 5, 6.*, a family of nested balls $\mathcal{B}_h(t_i)$is introduced. The $\mathcal{B}_h(t_i)$ functions are characterized by $h > 0$, and for a given h, $\mathcal{B}_h(t_i)$ represents the set of points for which the contribution of the VPL is significant. Said otherwise, the family of nested balls $\mathcal{B}_h(t_i)$ form a set of volumes in the 3D space, and for each $h > 0$, a single volume $\mathcal{B}_h$ exists. The geometric shape of $\mathcal{B}_h$ is a sphere whose center is constrained to (and passes through) a line(y_i, n_i), and has a diameter $D(h)$. h is a parameter of $\mathcal{B}_h$. For each point x outside of $\mathcal{B}_h(t_i)$, the transfer function $H(t_i, x, \overrightarrow{n_x})$ (discussed in reference to equations 6, 8, and 9) is smaller than h whatever the orientation of the receiver $\overrightarrow{n_x}$. This is expressed in equation 13

$$\forall h \in \mathbb{R}^*, \quad \mathcal{B}_h(t_i) = \left\{ x \in \mathbb{R}^3 \text{ s.t. } \max_{\vec{n}} H(t_i, x, \vec{n}) \geq h \right\}.$$

Furthermore, the transfer function $H(t_i, x, \overrightarrow{n_x})$ (discussed in reference to equations 6, 8, and 9) is maximal when the receiver is front facing the emitter, or said otherwise, $\overrightarrow{n_x} = \bar{x}y_i$. Thus, with VPL radiance distribution model, the $\mathcal{B}_h(t_i)$ functions can be written (equation 14):

$$\mathcal{B}_h(t_i) = \{x \epsilon \mathbb{R}^3 \; s.t. \frac{||x-y_i||}{\langle \overrightarrow{n_x}, \; \bar{x}y_i \rangle^+} \leq D(h) = \frac{1}{\pi}\sqrt{\frac{\rho_x \rho_i E(t_i)}{h}}\},$$

where $D(h)$ (the diameter of the sphere of $\mathcal{B}_h(t_i)$) represents the maximal distance of the contribution of the direct outgoing radiance of a VPL that belongs to a given subset As $D(h)$ does not depends on x, $(\mathcal{B}_h(t_i))_{h \epsilon \mathbb{R}^*}$ is a family of nested balls which frontier owns y_i and center lies on the line $(y_i \, , n_i)$.

The 3D partition of unity $(f^k)_k$ can be constant over the spheres being the frontier of a $\mathcal{B}_h(t_i)$. The $\mathcal{B}_h(t_i)$ function are the isolevel of the support functions. Then, by defining the following mapping from $\mathbb{R}^3$ to $\mathbb{R}$ (equation 15):

$$\forall x \epsilon \mathbb{R}^3, \qquad d(t_i, x) = \frac{||\bar{x}y_i||^2}{\langle \overrightarrow{n_x}, \; \bar{x}y_i \rangle^+}$$

the 1D partition of unity $(\tilde{f}^k)_k$that is obtained is defined by (equation 16)

$$\forall x \epsilon \mathbb{R}^3, \; f^k(t_i, x) = \tilde{f}^k(d(t_i, x)).$$

As partition of unity $(f^k)_k$will be used for performing a splatting operation, as known in the art. Hence, when the lighting of a VPL is computed for each pixel, an evaluation of each pixel is carried out for blending textures together on each pixel. During rendering, the partition of unity $(f^k)_k$may be made as smooth as possible while keeping them easy to compute and define them as (equation 17):

$$\forall d \in \mathbb{R}, \quad \tilde{f}^k(d) = \begin{cases} 1 & \text{if } k = 0 \text{ and } d \in]0, D_1] \\ \frac{d - D_{k-1}}{D_k - D_{k-1}} & \text{if } k > 0 \text{ and } d \in]D_{k-1}, D_k] \\ \frac{D_{k+1} - d}{D_{k+1} - D_k} & \text{if } k > 0 \text{ and } d \in]D_k, D_{k+1}] \\ 0 & \text{otherwise} \end{cases}$$

In addition, it is possible to define (equation 18)

$$\begin{cases} D_k = \sqrt{S_0 \mu^k}, \; \forall k \in [0 \ldots N] \\ D_{N+1} = D_N \end{cases}$$

so that a set of continuous piecewise affine functions with compact support is obtained. The so-defined functions are well suited if only if $D_N$is greater that the diameter of the scene, otherwise they may exists a pair of points in the scene that cannot contribute to light transport. This means that $D_N > R_{scene}$. This translates into the following condition on μ (Eqn. 4):

$$\mu > \sqrt[N]{4 \frac{\mathcal{R}^2_{scene}}{S_o}}.$$

Referring now to Figure 7, examples of visualization of the support functions $(f^k)_k$ that support the $\mathcal{B}_h(t_i)$ functions on a planar section. The $\mathcal{B}_h(t_i)$ function are the isolevel of the support function and define the contour of the support function. Interestingly, the distance (D_0, D_1, D_3) of the contribution of the sphere of $(\mathcal{B}_h(t_i))$ on its support function depends on the subset $(\mathcal{L}^0, \ldots \mathcal{L}^k, \ldots \mathcal{L}^N)$ so that the power of a VPL depends on the subset it belongs. Hence, when k gets closer to 0, $\mathcal{L}^k$ is more and more populated by triangles (and thus VPL) with less and less power, while when k gets closer to N, $\mathcal{L}^k$ contains less and less triangles (and thus VPLs) with more and more power. As shown on Figure 3 (f), low power VPLs are used to capture local lighting while high power VPLs are used to capture model distant lighting with crescent-shape support functions.

Figure 8 shows an example of a one dimensional partition of unity that are used for performing a splatting operation at rendering time for rendering a sphere of a $\mathcal{B}_h(t_i)$. The smoother is the function, the better are the computation results. The example of $\tilde{f}^k$ of figure 8 has good results as there are a limited numbers of overlaps (on the abscissa there are at the most two functions that are not zero-valued), the function is continuous and concave. $\tilde{f}^0$ is represented by the curve with the stars, $\tilde{f}^1$ with the triangles, $\tilde{f}^2$ with the dots, and $\tilde{f}^3$ with the diamonds on it. Several classes of functions can be used. In particular, a good base function of the set of support functions $(f^k)_k$ is a B-spline.

Finally, at step **S80**, the three-dimensional scene is rendered by lighting its set of triangles, the triangle of the subset of triangles $\mathcal{L}^*$ being used as indirect light sources according to their radius of influence. As mentioned, the support functions $(f^k)_k$ are used for carrying out splatting function for rendering the spheres of the $\mathcal{B}_h(t_i)$ functions of the family of nested balls. The rendering is performed as known in the art and uses the one or more direct light sources provided with the 3D scene for exploiting the computed VPLs.

Figures 10 and 11 are pictures (screenshots) of a 3D scene rendered with a global illumination. Figure 10 shows the scene rendered without the divergent pipeline: only the heavy tessellated geometry cast indirect lighting. Figure 11 shows the same 3D scene that is rendered by using the divergent pipeline; the ground

(which is comprised of 4 triangles) light bounce reveals much of the scene. Hence, the biggest triangles are those used that are in the subset(s) that is(are) the closer of $\mathcal{L}^N$ and that are used to model distant lighting.

For the sake of completeness, implementation details are discussed in reference to Figure 4. The pipeline only contains two geometry draw passes of the entire scene: one to generate the GBuffer and one to generate and splat the VPLs. These are the regular pipelines. This moderate use of the raw geometry is an important metric for application that require huge number numbers of polygons, for instance Computer-Aided Design (CAD) and the like software application. A typical CAD scene comprises thousands of 3D modeled objects in which each object comprises hundreds or thousands of triangles, and even more. In fact, a third geometry draw pass occurs, but involves only a fraction of the scene: the divergent triangles that are processed by the divergent pipeline. As the divergence criterion is set such that the number of divergent triangles remains small, this last pass is not computationally prohibitive. Indeed, the regular pipeline exploits intensively the geometry shader stage to perform computations on a per-triangle basis rather than a per-fragment or a per-vertex one. Interestingly, for recent GPU architecture, the formally prohibitive overhead of the geometry shader stage has been greatly reduced, enabling polygon-wise computations for large streams.

The divergent pipeline exploits hardware tessellation units are not designed to manage massive triangle sets as input, inducing a noticeable overhead while processing a triangle even if this triangle does not require any subdivision. At the same time, the geometry stage of these architectures are extremely efficient at discarding or letting passing through polygons, i.e. when no geometry amplification is mandatory. In the first pass of the regular pipeline the entire scene geometry (raw data) is processed but the tessellation stage is disabled. Divergent triangles (S320) are detected at the geometry stage and stored in a separate buffer, while regular ones are stochastically sampled (S520). In order to manage scenes with numerous materials and textures, a per-triangle material index may be stored, used in the following pass to fetch information from a material or texture atlas.

Graphic library provides features that allow optimizing rendering computations. For instance, using the OpenGL "glDrawArrayIndirect" feature, the divergent buffer on which non-regular triangles are stored is subsequently directly used as input geometry for the second pass without any CPU synchronization. The tessellation stage is solely activated for this particular pass and used to subdivide (S322) the triangles such that their area becomes small enough to be processed by our regular pipeline. In general, the number of large triangles is relatively small compared to the total triangle count. Consequently, the overhead induced by the divergent buffer filling and vertex processing is negligible.

The VPL support functions $(f^k)_k$can be made as smooth as possible; this allows producing at each frame a visually plausible rendering without high frequency artifacts. Nevertheless, with the aforementioned global uniform variable u_rand, it is straightforward to generate many independent renderings of the same scene. In addition, for each triangle, one can easily derive two new independent random values from u_rand. These values are used to jitter the VPL center y_i defined in equation 6 over the triangle t_i.

Then, by averaging all these renderings with jittered VPLs, equation 9 provides a result that is allows the computation of the first bounce outgoing radiance on surface composed of diffuse reflectors and ignoring indirect visibility. While this means that indirect lighting is computed by integrating over every scene triangle, this progressive version of the present method is able to manage any kind of disturbed geometry (e.g. normal mapping, alpha tested) and could also be extended to manage emissive textured geometry.

The way the VPL contributions are summed to simulate the indirect lighting is orthogonal to previous discussion. Hence, to keep a simple pipeline, a splatting strategy similar to deferred lighting discussed in *SAITO, T., AND TAKAHASHI, T. 1990. Comprehensible rendering of 3-d shapes. In ACM SIGGRAPH Computer Graphics, vol. 24, ACM, 197–206 may be used.* In particular, this allows to manage geometry decimation, VPLs generation and lighting in a single shader program. Indeed, besides determining whether a triangle is divergent or not and which is the level of generated VPLs, the geometry shader can be used to transform input triangles in sized point

primitives which encompasses the underlying VPL screen space function support. Undersampling the resulting signal may be a good optimization. To do so, the viewport is partitioned in $4^{Ntiling\ level}$ tiles and assign to each pixel in the viewport a unique tile pixel at the same relative location – this technique is often referred as interleaved sampling (*KELLER, A., AND HEIDRICH, W. 2001. Interleaved sampling. Springer.*). Next, at splatting time, a tile is randomly chosen for each VPL thus dividing the number of touched pixels by $4^{Ntiling\ level}$. The image is recomposed by untiling the buffer and blurring it to remove the generated noise.

CLAIMS

1. A computer-implemented method for rendering the global illumination of a three-dimensional scene, comprising:

- providing (S10) a three-dimensional scene that comprises of a set $\mathcal{L}$ of triangles and one or more direct light sources;
- determining (S30) that each triangle t_i of the set has an area that is below a threshold S_o;
- assigning (S50) to each triangle of the set a radius of influence using a probability law;
- obtaining a subset of triangles $\mathcal{L}^*$ by filtering out (S510) the triangles according to their radius of influence;
- rendering (S80) the three-dimensional scene by lighting its set of triangles, the triangle of the subset of triangles $\mathcal{L}^*$ being used as indirect light sources according to their radius of influence.

2. The computer-implemented method of claim 1, wherein determining that each triangle of the set has an area that is below the threshold comprises:

- partitioning the set of triangles $\mathcal{L}$ into:
    - a first set of triangles (S310), wherein each triangle of the first set has an area that is below the threshold;
    - a second set of triangles (S320), wherein each triangle of the second set has an area that is above the threshold; and
- tessellating (S322) each triangle of the second set into a third set of triangles, wherein each triangle of the third set has an area that is below the threshold as a result of the tessellation;

and wherein

- assigning to each triangle of the set a radius of influence using a probability law comprises assigning to each triangle of the first and the third sets a radius of influence using a probability law; and

- obtaining a subset of triangles comprises obtaining a subset of triangles by filtering out the triangles of the first and third sets according to their radius of influence.

3. The computer-implemented method of one of claims 1 to 2, further comprising, after determining that each triangle of the set has an area that is below a threshold:
- randomly partitioning (S50) the set of triangles into $N + 2$ subsets of triangles according to the probability law;
- discarding (S510) one of the $N + 2$ subsets of triangles that comprises triangles that are determined as having a low contribution to the rendering of the three-dimensional scene, the remaining $N + 1$ subsets $\mathcal{L}^*$comprising non-discarded triangles.

4. The computer-implemented method of claim 3, wherein the remaining subsets $\mathcal{L}^*$of the $N + 2$ subsets of triangles comprise triangles t_i of the set such that

$$\forall t_i \in \mathcal{L},\ P(t_i \in \mathcal{L}^*) = \frac{\mathcal{A}(t_i)}{\mathcal{S}_0},$$

wherein $A(t_i)$ is the surface area of the triangle t_i and S_o is the threshold.

5. The computer-implemented method of claim 4, wherein each triangle of the remaining subsets $\mathcal{L}^*$is dispatched (S520) in one of the remaining subsets $\mathcal{L}^k$ according to a probability for a triangle to lie in a subset $\mathcal{L}^k$ defined by:

$$P(t_i \in \mathcal{L}^k) = \begin{cases} \frac{\mathcal{A}(t_i)}{\mathcal{S}_N}, & \text{if } k = N \\ \frac{\mathcal{A}(t_i)}{\mathcal{S}_k} - \frac{\mathcal{A}(t_i)}{\mathcal{S}_{k+1}}, & \text{if } 0 \leqslant k < N \end{cases}$$

wherein the value of S_k increases for when k increases.

6. The computer-implemented method of claim 5, wherein S_k is set such that $S_k = S_o\mu^k$, where $\mu > 1$ and the value of μ is a real number comprised between [1; 5].

7. The computer-implemented method of claim 5, wherein S_k is set such that $S_k = S_o\mu^k$, wherethe value of μ is defined by $\mu > \sqrt[N]{4\frac{\mathcal{R}_{scene}^2}{S_o}}$ with $\mathcal{R}_{scene}$that is the radius of the three-dimensional scene.

8. The computer-implemented method of one of claims 5 to 7, further comprising, after that each triangle of the remaining subsets $\mathcal{L}^*$has been dispatched:
- computing (S60) a single virtual point light for each triangle of the remaining subsets $\mathcal{L}^*$;
and wherein assigning to each triangle of the set a radius of influence using a probability law further comprises:
- computing an outgoing radiance (S70) for the virtual point light of each triangle of the remaining subsets $\mathcal{L}^*$, the computation being performed according to the subset in which the triangle has been dispatched.

9. The computer-implemented method of claim 8, wherein computing an outgoing radiance for the virtual point light of each triangle comprises:
- computing a family of nested balls $\mathcal{B}_h(t_i)$ representing the set of points in the three-dimensional scene for which the contribution of the virtual point lights is significant;
- computing, for each remaining subset of the $N + 2$ subsets, a support function $f^k(t_i, \mathrm{x})$ such that the set of support functions $(f^k)_k$ over the remaining subsets form a partition of unity that is constant over a frontier of a $\mathcal{B}_h(t_i)$ of the family of nested balls.

10. The computer-implemented method of claim 9, wherein a base function of the set of support functions $(f^k)_k$is a B-spline.

11. The computer-implemented method of claim 9 or 10, wherein $\mathcal{B}_h(t_i)$ is defined by $\mathcal{B}_h(t_i) = \{x\epsilon\mathbb{R}^3\ s.t.\frac{||x-y_i||}{\langle\overrightarrow{n_x},\ \bar{x}y_i\rangle^+} \leq D(h) = \frac{1}{\pi}\sqrt{\frac{\rho_x\rho_i E(t_i)}{h}}\}$,
wherein x is a point of the three-dimensional scene with normal $\overrightarrow{n_x}$,

y_i is the point on which the virtual point light is centered on the triangle t_i,

ρ_x is albedo at point x,

ρ_i is albedo at point y_i,

$E(t_i)$ is a direct irradiance falling to the triangle t_i.

12. The computer-implemented method of one of claims 9 to 11, wherein, at the rendering step, the set of support functions $(f^k)_k$ is used for carrying out splatting operations when rendering a sphere of a $\mathcal{B}_h(t_i)$ of a family of nested balls.

13. The computer-implemented method of one of claims 1 to 12, wherein the threshold S_o is defined by $S_o = 4\pi \frac{D_{near}^2}{N_{Avg}}$,

wherein $N_{Avg}$is an average number of virtual point lights;

D_{near} is a minimal distance between pixels of the 3D scene and VPLs.

14. The computer-implemented method of one of claims 1 to 13, further comprising, before determining that each triangle t_i of the set has an area that is below a threshold S_o:

- generating (S20) a single random integer for each vertex of triangle t_i of the set.

15. The computer-implemented method of claim 14 combined with claim 2, wherein computing a single random integer for each vertex of triangle further comprises:

- for each sub-vertex of a triangle of the third set, determining a barycentric coordinate of the sub-vertex;
- using the barycentric coordinate to fetch (S210) a random value in a noise texture;
- using the fetched random value to generate (S220) the single random integer of the said each sub-vertex of a triangle of the third set.

16. A graphic library computer program comprising instructions for performing the method of any of claims 1-15.

17. A system comprising a processor coupled to a memory and a graphical card with a graphic processing unit, the memory having recorded thereon the computer program of claim 16.

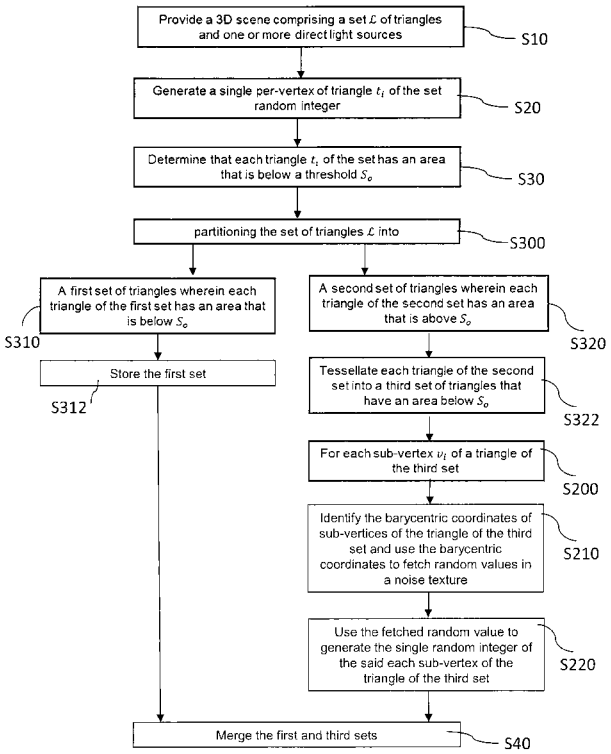
18. The system of claim 17, wherein the graphical card comprises:
- a geometry shader unit for determining that each triangle t_i of the set has an area that is below a threshold S_o; and
- a tessellation unit for tessellating triangles t_i of the set;
and wherein the computer program recorded on the memory is adapted to configure the tessellation unit for performing the generation of the single random integers according to claim 14 or 15.

ABSTRACT

The invention notably relates to a computer-implemented method for rendering the global illumination of a three-dimensional scene. The method comprises providing a 3D scene that comprises of a set of triangles and one or more direct light sources, determining that each triangle of the set has an area that is below a threshold, assigning to each triangle of the set a radius of influence using a probability law, obtaining a subset of triangles by filtering out the triangles according to their radius of influence, rendering the three-dimensional scene by lighting its set of triangles, the triangle of the subset of triangles being used as indirect light sources according to their radius of influence.

Figure 3

【図１】


Provide a 3D scene comprising a set $\mathcal{L}$ of triangles and one or more direct light sources
S10
Generate a single per-vertex of triangle t_i of the set random integer
S20
Determine that each triangle t_i of the set has an area that is below a threshold S_o
S30
partitioning the set of triangles $\mathcal{L}$ into
S300
A first set of triangles wherein each triangle of the first set has an area that is below S_o
S310
A second set of triangles wherein each triangle of the second set has an area that is above S_o
S320
Store the first set
S312
Tessellate each triangle of the second set into a third set of triangles that have an area below S_o
S322
For each sub-vertex v_i of a triangle of the third set
S200
Identify the barycentric coordinates of sub-vertices of the triangle of the third set and use the barycentric coordinates to fetch random values in a noise texture
S210
Use the fetched random value to generate the single random integer of the said each sub-vertex of the triangle of the third set
S220
Merge the first and third sets
S40

【図２】

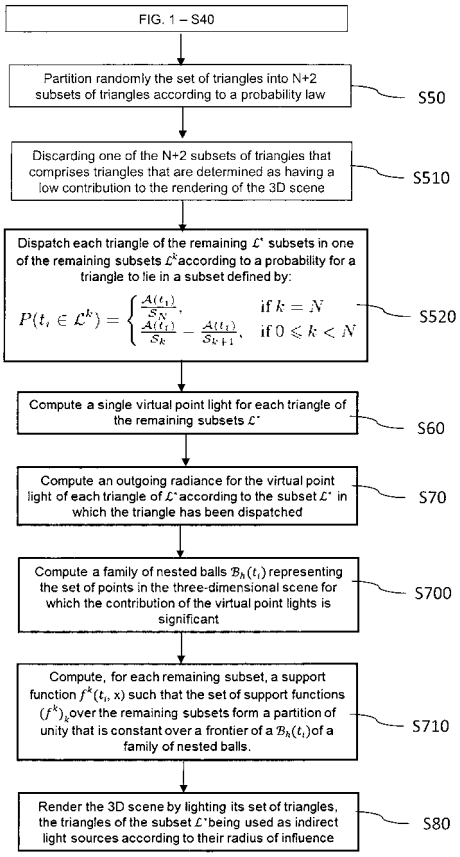
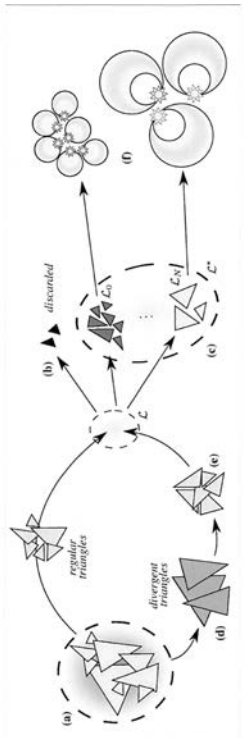
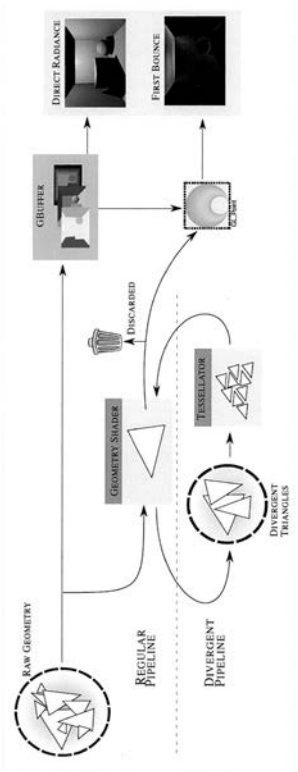


FIG. 1 – S40
Partition randomly the set of triangles into N+2 subsets of triangles according to a probability law
S50
Discarding one of the N+2 subsets of triangles that comprises triangles that are determined as having a low contribution to the rendering of the 3D scene
S510
Dispatch each triangle of the remaining $\mathcal{L}^*$ subsets in one of the remaining subsets $\mathcal{L}^k$ according to a probability for a triangle to lie in a subset defined by:
$P(t_i \in \mathcal{L}^k) = \begin{cases} \frac{\mathcal{A}(t_i)}{S_N}, & \text{if } k = N \\ \frac{\mathcal{A}(t_i)}{S_k} - \frac{\mathcal{A}(t_i)}{S_{k+1}}, & \text{if } 0 \leqslant k < N \end{cases}$
S520
Compute a single virtual point light for each triangle of the remaining subsets $\mathcal{L}^*$
S60
Compute an outgoing radiance for the virtual point light of each triangle of $\mathcal{L}^*$ according to the subset $\mathcal{L}^*$ in which the triangle has been dispatched
S70
Compute a family of nested balls $\mathcal{B}_h(t_i)$ representing the set of points in the three-dimensional scene for which the contribution of the virtual point lights is significant
S700
Compute, for each remaining subset, a support function $f^k(t_i, x)$ such that the set of support functions $(f^k)_k$ over the remaining subsets form a partition of unity that is constant over a frontier of a $\mathcal{B}_h(t_i)$ of a family of nested balls.
S710
Render the 3D scene by lighting its set of triangles, the triangles of the subset $\mathcal{L}^*$ being used as indirect light sources according to their radius of influence
S80

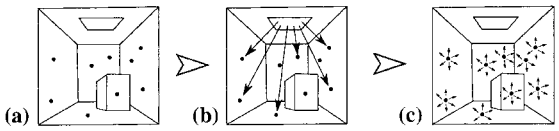
【図３】


discarded
regular triangles
divergent triangles

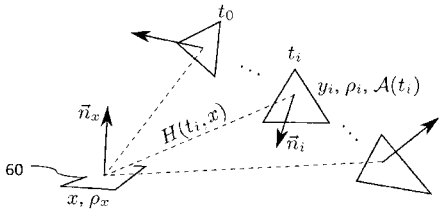
【図４】


DIRECT RADIANCE
FIRST BOUNCE
GBUFFER
DISCARDED
TESSELLATOR
GEOMETRY SHADER
DIVERGENT TRIANGLES
RAW GEOMETRY
REGULAR PIPELINE
DIVERGENT PIPELINE

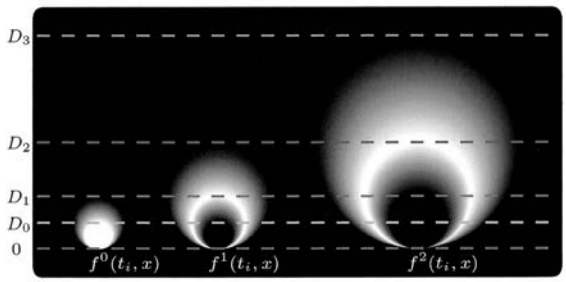
【図５】



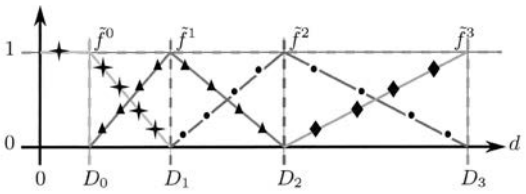
【図６】



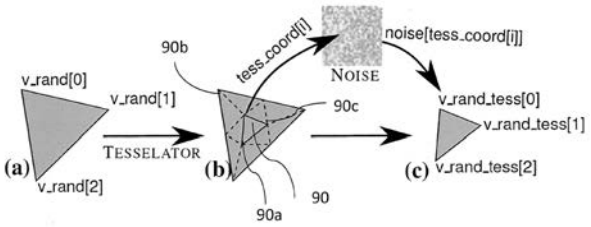
【図７】



【図８】



【図９】



【図１０】

【図１１】

【図１２】