JAERI-Research
97-062

モンテカルロ法を適用した光線追跡画像生成プログラムの開発

1997年9月

樋口健二・大谷孝之・長谷川幸弘*

日本原子力研究所
Japan Atomic Energy Research Institute

本レポートは，日本原子力研究所が不定期に公刊している研究報告書です。
入手の問合わせは，日本原子力研究所研究情報部研究情報課（〒319-11 茨城県那珂郡東海村）あて，お申し越しください。なお，このほかに財団法人原子力弘済会資料センター（〒319-11 茨城県那珂郡東海村日本原子力研究所内）で複写による実費頒布をおこなっております。

This report is issued irregularly.
Inquiries about availability of the reports should be addressed to Research Information Division, Department of Intellectual Resources, Japan Atomic Energy Research Institute, Tokai-mura, Naka-gun, Ibaraki-ken, 319-11, Japan.

編集兼発行　日本原子力研究所
印　　刷　いばらき印刷㈱

モンテカルロ法を適用した光線追跡画像生成プログラムの開発

日本原子力研究所計算科学技術推進センター
樋口　健二・大谷　孝之・長谷川幸弘*

(1997年8月12日受理)

光線追跡法は、3次元画像を生成するための強力な手法である。従来の光線追跡法による画像生成においては、視点を始点としてスクリーンを構成する各画素を通過する半直線、すなわち視線を追跡することで各画素に対する輝度値を計算する。

しかし、この場合、(1) 光源の幾何形状表現が困難である、(2) 散乱反射の計算に大まかな近似を用いざるを得ない、といった理由から生成される画像の現実感が大きく失われる。

そこで、筆者らは、原子力分野で広く用いられているモンテカルロ法により、視線ではなく光線を追跡する順方向光線追跡プログラムを開発した。この手法により、上記問題点を解消するとともに、粒子輸送モンテカルロ・コードの分散低減法を適用し、さらに高速モンテカルロ装置を利用することで計算時間を大幅に短縮した。

日本原子力研究所計算科学技術推進センター：　〒153　東京都目黒区中目黒2－2－54
＊（財）高度情報科学技術研究機構

Development of Ray Tracing Visualization Program by Monte Carlo Method

Kenji HIGUCHI, Takayuki OTANI and Yukihiro HASEGAWA*

Center for Promotion of Computational Science and Engineering
Japan Atomic Energy Research Institute
Nakameguro, Meguro-ku, Tokyo

(Received August 12, 1997)

Ray tracing algorithm is a powerful method to synthesize three dimensional computer graphics. In conventional ray tracing algorithms, a view point is used as a starting point of ray tracing, from which the rays are tracked up to the light sources through center points of pixels on the view screen to calculate the intensities of the pixels.

This manner, however, makes it difficult to define the configuration of light source as well as to strictly simulate the reflections of the rays.

To resolve these problems, we have developed a new ray tracing means which traces rays from a light source, not from a view point, with use of Monte Carlo method which is widely applied in nuclear fields. Moreover, we adopt the variance reduction techniques to the program with use of the specialized machine (Monte-4) for particle transport Monte Carlo so that the computational time could be successfully reduced.

Keywords: Computer Graphics, Ray Tracing Method, Monte Carlo Method, Simulation

*Research Organization for Information Science & Technology

目　　次

Contents

1. はじめに

一般に、境界表現法や体積表現法等の手法によって表現された三次元物体の可視化手法は、以下の３つに大別することができる。

・ポリゴン・データのスクリーン上へのアフィン変換による画像生成手法
・ラジオシティ法
・光線追跡法

これらの手法はそれぞれ次のような原理によって三次元画像が生成される。

(1) ポリゴン・データのスクリーン上へのアフィン変換による画像生成手法

この手法は、複数のポリゴンの集合体として表現された三次元物体をスクリーン上にアフィン変換（投影）し、さらに各ポリゴンに対して濃淡づけのための輝度計算を行うことによって、三次元画像を得る手法である。この際、最終的にスクリーンを構成する各画素に投影されるポリゴンを効率良く決定するための代表的手法として、スキャンライン・アルゴリズムとZ-バッファ・アルゴリズムがある [1, 2]。

スキャンライン・アルゴリズムによる画像生成法は、次の手順によってなされる。(a) スクリーン上に投影された各物体を構成する全てのポリゴンとスクリーンを縦方向にスキャンするラインとの交差判定およびポリゴン同士の重なり具合いの判定により、スクリーンを構成する各画素を占有するポリゴンおよびポリゴン上の点を決定する。(b) ポリゴン上の点に対する輝度値を求め、これを各画素の輝度とする。(c) 以上の手続きを全てのスキャン・ラインに対して行う。

一方、Z-バッファ・アルゴリズムによる画像生成は、各ポリゴンに対して、以下の処理を行うことによってなされる。(a) ポリゴンをスクリーン上に投影する。三次元空間内のポリゴンをスクリーンに対応する二次元平面にアフィン変換した時、そのポリゴンが、スクリーンの外にある場合、処理を終了する。(b) 視点から、そのポリゴンまでの距離を計算し、その距離が、既にデプス・バッファ[1]に書き込まれた距離より短い時、デプス・バッファにその値を書き込み、次の処理に移る。デプス・バッファに書かれた距離より長い時は、そのポリゴンに対する処理を終了する。(c) ポリゴンがスクリーン上で占有する画素に対する輝度を計算し、その値をフレーム・バッファ[2]に書き込む。

しかし、これらポリゴンデータからの画像生成では、環境中の放射光や反射光の相互作用などを計算することはなく、写実性の面ではラジオシティ法や光線追跡法におよばない。

(2) ラジオシティ法

ラジオシティ法では、定常状態における閉領域に関するエネルギー・バランスの式を解く

[1]デプス・バッファ：各画素に対する距離情報が、書き込まれたバッファ。初期値として、無限遠点に相当する値が書き込まれている。

[2]フレーム・バッファ：各画素に対する輝度値が、書き込まれたバッファ。

[3]。体系内の各領域（可視化される物体）は、複数のパッチ（面）によって構成される。光源およびスクリーンもその表面は複数のパッチによって覆われている。パッチの数を増やすことで解の精度は高まる。各パッチに対して、その面から発したエネルギーが他の面に到達する割合を示すフォームファクタと呼ばれる値を設定後、パッチ相互間のエネルギーの出入りの式をたてる。

これらの連立方程式を解き、各パッチからスクリーンを構成する各画素へのラジオシティ(各パッチから発する光）を求めることで画像を生成する。環境を構成するすべての面からの放射光や反射光の相互作用を計算するため現実感のある画像の生成が可能である。ただし、可視化する体系のモデリングに極めて多大の作業を必要とすることが実用上の問題点である。

(3) 光線追跡法

光線追跡法によるプログラムは、Zバッファ・アルゴリズムやスキャンライン・アルゴリズムと比較して構造が簡単で、しかも、より写実性に優れた三次元画像を得ることが可能である [4, 5]。また、光の透過現象を扱うことが可能である、という長所を持つ。多大の計算時間を要するという欠点を持つが、各光線（画素）に対する計算は独立であることから、ベクトル計算あるいは並列計算による高速化が可能である。ゆえに、高並列あるいは超並列といった計算機が開発されつつある計算機開発の現状を鑑みれば、いま最も興味深い画像生成手法であるといえる。

従来の光線追跡法による画像生成においては、視点を始点としスクリーンを構成する各画素を通過する半直線、すなわち、視線を追跡することで各画素に対する輝度値を計算する、いわゆる逆光線追跡法であった。しかし、この場合、(1) 光源の幾何形状表現が困難である、(2) 散乱反射および鏡面反射のシミュレーションを物質と光の入射角度等によって比例配分あるいは変化させる、といった理由から生成した画像は人工的な質感を持ち、現実感に乏しいものであった。

これを改善するアプローチとして、二つの代表的な手法が挙げられる。一つは、Cook らによる分散型光線追跡法であり、いま一つは、Pattanaik らによる順方向光線追跡法である。

Cook らは、視線追跡法により「ぼやけた感じ」を出すために、追跡する視線を時間方向に分割した。すなわち、空間方向ではなく、時間方向の分割により Fuzzy な現象を扱うことのできる手法を提案した。Cook らは、この手法により、動く物体が作るブレ (motion blur)、線光源や面光源が作り出すぼやけた濃淡、カメラや人間の目が持つ視野範囲 (depth of field) によって作り出される「ぼやけた感じ」を表現した [7]。

Pattanaik らは、「複雑な 3 次元環境のいかなる表面上の点の照度もその点からの光の放出によるものであり、その点を中心とする半球方向から入射した光の反射によるものである」という基本的な概念に基づいて、環境中のすべての点が持っている、他の点への照射に寄与するポテンシャルを表現する方程式をたて、これを順方向の粒子追跡によって解くことで画像が生成されることを示し、その際にモンテカルロ法を適用したサンプリングによって計算の効率を向上できることを示した [6]。しかし、Pattanaik らの成果は簡単なモデルにおいて例証されているのみで、実際に生成される画像の写実性の評価等はなされていない。

本研究は、Pattanaikらと同様に、光源からの光線を追跡し、光の反射をシミュレーションすることで上記問題点を解消し、自然な画像を生成するものである。光（粒子）の伝播シミュレーションにはモンテカルロ法を用いており、原子力分野で広く用いられている粒子輸送モンテカルロ・コードの分散低減法を利用することで計算時間の短縮を図っている。また、同じく粒子輸送モンテカルロ・コードの高速化を目的として開発された高速モンテカルロ装置により画像生成処理を高速化している。開発したコードは、RTMC(Ray Tracing program by Monte Carlo method) と名付けられた。

開発したシステムの性能評価および従来手法との比較には、実規模の原子力施設、すなわち日本原子力研究所の研究用原子炉JRR-3Mの一次区画（原子炉からの一次冷却水を浄化したり、実際に原子炉から放出される熱利用に供する二次系との熱交換を行う装置類が設置された区画）内主要機器の可視化を行った。

2. 光線追跡法の原理と計算の特徴

2.1 光線追跡法の原理

光線追跡法の原理は、人間の目（あるいはカメラ等の撮像機）に情景が映る過程を模擬することにより、画像を生成することであり、これに拠るプログラムにおいては一種のシミュレーションが行われている。つまり、ある情景が人間の目に映るということは、物体によって反射された光源（太陽光や室内光）からの光が人間の目に届くということであり、光線追跡法による画像生成プログラムにおいては、光源から出た光線を追跡し、体系中のどの物体と衝突するかを決定し、拡散／鏡面反射および屈折現象によって方向を変えた光線をさらに追跡する、というシミュレーションを行う。

このシミュレーションにより、体系中に置かれたスクリーン上の各画素を通過し目に到達する光線の強度を求め、各画素に対するRGB値を決定することにより、三次元画像を得る。

ところが、前章でも述べたように、従来の光線追跡法では計算時間の短縮のために光線を逆方向から追跡し、間接光による拡散反射光については一定値の環境光を用いて誤魔化し拡散反射に寄与するすべての光線を追跡することはしていない。本研究では、これが画像の写実性をそこなう原因であると仮定し、原理に忠実にシミュレーションを行うことで写実性の優れた画像生成システムを開発する。

光線を光源方向から追跡することによって追跡すべき光線数が大きくなる点については、ベクトル計算手法および並列計算手法を用いてこれに対処し、間接光による拡散反射の影響については、光線が拡散反射面と交点を持ったならばその点で四方八方に飛び散る光線を生成すればよい。したがって環境中の各点の照度計算には、この四方八方に散らばった膨大な光線の寄与の積分計算が不可欠になる。しかし、このとき、必ずしもその膨大な数の光線が必要になるわけではないことが、モンテカルロ法による数値積分法から推察される[7]。モンテカルロ法によるn次元の数値積分法によれば、普通の積分法でそれぞれの次元方向にN等分して得られる積分値（すなわち計算点はN^n点）と、乱数によりn次元空間の計算点をN個発生して推測される積分値がほぼ同程度の精度を有することが示されている。

したがって、Pattanaikらのように拡散相互反射の計算にモンテカルロ法を適用する発想が生まれ、すると、原子力分野でよく知られている放射線（粒子）輸送問題を扱うモンテカルロ法と順方向の光線追跡法とが酷似していることに気付く。

2.2 粒子輸送モンテカルロ・コード

モンテカルロ法は、原子力分野において中性子や光子の輸送問題の数値解法として、各種の実験解析、臨界安全解析などに広く用いられている。モンテカルロ法は、母集団から代表粒子をサンプリングし、粒子の振る舞いを模擬することによって物理量の計算を行う手法で、

たとえば、原子力分野の遮蔽あるいは臨界計算においては、粒子が媒質を構成する核種との反応によって吸収されたり、あるいは体系外へ漏れたりするまでの粒子の振る舞いを模擬することによって粒子束、増倍率などの計算を行う。

粒子輸送モンテカルロ・コードの計算フローは、扱う粒子の種類や統計値として求める物理量に応じて若干異なるが、大要は粒子を１個ずつ取り出して以下のような処理を行うものである。

(a) 乱数と分布関数等粒子源に対する条件を用いて、粒子の初期値（位置、飛方向、エネルギー等）を決定する。

(b) 粒子の現在位置から飛方向に沿った領域の境界までの距離を計算する。

(c) 境界までの距離と粒子のパスの長さを比較し、粒子が領域の境界を越えたかどうかをチェックする。

(d) Flux や Energy deposition 等粒子のその領域への寄与分を計算する。

(e) 粒子が領域の境界を越えた場合、粒子が次に進入する領域を探し、(b) へ戻る。ただし、粒子が体系外へ出た場合は、この粒子の追跡を終了する。粒子が、その領域にとどまった場合は、(f) において衝突のシミュレーションを行う。

(f) 粒子と領域の衝突点において、乱数と断面積データを用いて、衝突のシミュレーションを行う。このとき、粒子が散乱したか、二次粒子を生成した場合、さらに追跡を続けるため (b) へ戻る。粒子が吸収された場合は、追跡を終了する。

2.3 粒子輸送問題のためのモンテカルロ法と光線追跡法との類似点

光の散乱過程を模擬している順光線追跡法による画像生成プログラムの構造および処理は、放射線輸送問題を扱うモンテカルロ・プログラムのそれらと酷似している。この理由は、両者は、複雑な幾何形状によって表現された三次元空間において、粒子（放射線）あるいは光線を追跡するシミュレーション・プログラムであるという点にある。

放射線輸送問題を扱うモンテカルロ・プログラムと光線追跡法による画像生成プログラムとの対応づけは、線源＝光源、放射線＝光線、原子核と衝突後の中性子の散乱および吸収＝物体と衝突後の光線の拡散／鏡面反射および屈折、検出器＝スクリーン上の各画素、放射線のエネルギー＝光の強度、となる。これらの物理現象のモデル化およびそれに対応する両者の処理プログラムは酷似しており、さらに、モンテカルロ・プログラムにおいて、より良い統計的精度を得るために粒子数（追跡する放射線の数）を増やす点も、光線追跡法において、より現実感のある画像を得るために光線の数を増やすことと同じ意味を持つ。放射線輸送問題を扱うモンテカルロ・プログラムと光線追跡法の比較を Table2.1に示す。

また、放射線輸送問題を扱うモンテカルロ・プログラムにおいては、原子核と衝突後の放射線の振る舞い等が乱数と確率（断面積）を使って決定されるのに対し、従来の逆方向光線追跡法による画像生成プログラムにおいては物体と衝突後の光の拡散／鏡面反射および屈折の

比率等が予め与えられた一定の割合によって決定する点が大きく異なっていたが、本作業で作成するプログラムでは、光の拡散／鏡面反射のシミュレーションもモンテカルロ法によって行う。このため、従来の画像生成法で広く用いられているRGB値による環境中の物体表面色の表現方法は適用できず、光の波長ごとの反射率として再定義する必要がある。

Table 2.1 Comparison between Monte Carlo code for particle transport and ray tracing method.

手法 処理	粒子輸送モンテカルロ法	レイトレーシング
粒子の取り出し	線源	光源
粒子の種類	中性子、光子、電子	光線
衝突解析	粒子の散乱、吸収	光の拡散・鏡面反射
統計媒体	検出器、領域、面	スクリーン上の画素
エネルギー	粒子のエネルギー	光の強度

3. 画像生成アルゴリズムと計算モデル

順方向光線追跡法による画像生成アルゴリズムについて述べる。光線追跡法においては、光源から出た光線を追跡し、スクリーンを構成する各画素を通過し目に到達する光線の強度（RGB値の輝度値）を決定することで、画像が生成される。物体と衝突後の光線は、拡散／鏡面反射および屈折のシミュレーションの結果に応じて新たな光線を発生する。画像生成プログラムの計算フローをFig.3.1およびFig.3.2に示す。また、本プログラムにおける計算モデルを以下に示す。

3.1 投影と色づけ

本プログラムにおける像の描画方式は、コンピュータ・グラフィックスが発達してきた初期に提案され、現在も一般に広く用いられている投影法を用いている。

投影法はFig.3.3に示すように、ある空間に１つの平面である投影面（スクリーン）を設定する。次にこの投影面に含まれない１点を設定し、これを視点と呼ぶ。視点から空間の点に向かう半直線を視線という。この視線と投影面の交点を求めることを点の投影といい、この操作を空間内の３次元形状の各点について行うことを３次元形状を投影するという。

このようにして投影面に投影された２次元像は、ちょうど、視点のところにピンホールカメラを置いた際に得られる画像と相似である。したがって、投影法を用いることによって、人間の目に映る像を再生することができる。ただし、被写界深度は考慮されていない。被写界深度とは、レンズのピントが合う範囲のことを指し、人間の目では焦点からある程度（被写界深度）以上離れた物体はぼやけて見えるが、投影法による画像生成ではすべての物体についてピントがあった画像が生成される（画像がぼけない）。

投影法についての説明は、従来手法のほうがわかりやすいため、Fig.3.3には、視線追跡法の例を示したが、本手法の場合も投影の原理は同じである。ただし、本手法の場合、物体表面で反射し、スクリーン上のある画素を通り (たまたま) 視点に到達した光の寄与分のみを勘定しているため、「投影」というよりも「結像」という記述のほうが適切であろう。

色については、光、物体、視覚の３つの要素を考慮する必要がある。色は、まず光がなければ見ることができない。暗闇では色は存在し得ない。しかし、光だけで色が見えるわけではなく、物体に色がなければならない（厳密には物体表面が特定の波長の光を反射する性質をもつことを意味するが、これについては後述する）。最後に、人間の視覚が必要になる。光に照らされた物体からは、光が反射したり透過したりして人間の目に入るがこの光の構成が物体の色の種類によって異なっていて、種々の色として視覚で区別される仕組みになっている。

我々の周囲にある多くの物体は、何らかの色をもっているように見えるが、それらの物体そのものが色をもっているわけではない (発光しているわけではない)。赤いリンゴを例にとれば、リンゴ自体が色をもっているわけではなく、リンゴの表面が可視光線のうち超波長側

の光を多く反射し、短波長側の光を多く吸収するために、人間の目にはそれが赤く見えるにすぎない。つまり、種々の物体の色は可視光線のどの部分を反射し、相対的にどの部分を吸収するかによって決定づけられる。したがって、物体表面に波長毎の反射・吸収特性を与えておき、光を各波長毎に追跡することによって、物体の色を生成することができる。従来の視線方向からの光線追跡画像生成では、仮想的に色をもった物体をモデリングしているが、本プログラムでは光源の波長分布を定義し、これを波長毎に追跡することにより物理的に忠実な色の決定方法を採用している。

3.2 光源

本プログラムにおいては、光源から追跡する光線をサンプリングする。光源の幾何形状データ（強度分布を示す配光曲線）と一様乱数を用いて、光線の位置（始点）、方向をサンプリングする。サンプリング数は、領域の複雑さに応じて可変とする。また、一般に光線は複数波長の光の集合で、連続的な波長分布をもっているが、ここでは、各光線は離散的な波長の強度分布を属性としてもつ。

では、連続的な波長分布をどの程度細分化してサンプリングすればよいかであるが、ここではヤングーヘルムホルツの３原色説をとりあげる。これは、ヤングによって提唱され、ヘルムホルツによって体系化されたもので、「人間の網膜内には３つの基本色（赤、緑、青）に対して興奮する３つの視物質が存在し、それらの視物質の興奮する割合に応じて色覚を生ずる」という説である。赤、緑、青の３つの色光を適当に組み合わせることによって、人間の目に見える色が作り出せることはよく知られているが、３原色説が正しければ、この現象はきわめて容易に理解できる。1963 年、マークス [8] は、金魚の網膜から１個の錐体（網膜上の視細胞の１種）を取り出し、350～750nm の単色光の微少光点を当て、どの透過光を測定することによってそれぞれ赤、緑、青の波長を最もよく吸収する３種類の錐体の存在を発見し、３原色説が正しかったことを証明した。また、富田ら [9] は、鯉の網膜の錐体に微小電極を刺し、単色光のフラッシュを連続的に与えて錐体の電気的な反応を記録する方法で同様の３種の錐体の存在を証明した。その後、種々の方法により、人間の網膜にもそれぞれ 440、540、580nm 付近の波長の光に最良の感度をもつ３種の錐体が存在することが確かめられている。

本プログラムでは、上記および色彩は最終的に RGB 表現としてモニタ（あるいはプリンタ）上に表示されることから、光源から発する光を３つの波長に分解し、物体表面の色彩を３つの波長に対する反射率としてモデル化する。本プログラムで使用する波長群と対応する色を Table3.1 に示す。

また、本プログラムで扱う光源の幾何形状は、平行光線、線光源、点光源のみとする。平行光線とは、太陽光のような無限遠点からの光であり、この場合、体系内での光線の角度および光のエネルギー（強度）は一定であると考える。つまり、光のエネルギー（強度）が光源からの距離の二乗に反比例して減少するという法則は考慮に入れない。また、線光源や点光源において配光曲線が定義されている場合の光線のサンプリングには、面の法線と光源との角度の関数として定義された光の強度を使用する。

3.3 反射および透過

本プログラムにおいては、光の衝突過程、すなわち、拡散／鏡面反射および透過をモンテカルロ・シミュレーションする。

各光線は統計的な重要度を示すウェイト w を持ち、w の初期値は、1.0 である。衝突を起こす度に下記に示す吸収率に比例してその値は減少する。すなわち、衝突における吸収量を $\sum a$ とすると、$w = w \times (1 - \sum a / \sum t)$ である（$\sum t$ はトータル）。ウェイトが減少し、一定のしきい値 w_{th} 以下になった光線については、一様乱数を用いてルシアン・ルーレット（粒子輸送モンテカルロ・コードで用いられている分散低減法）にかける。ルシアン・ルーレットの結果生き残った光線については、その値を w_{av} で表現される平均値にウェイトを設定し直す。

スクリーンを構成する各画素に対する寄与分は、$w \times E$（光線のエネルギー）によって計算される。吸収、反射、透過の関係は次のとおり。

$$\sum t = \sum s + \sum a + \sum tr \tag{3.1}$$

$$\sum s = \sum e + \sum i \tag{3.2}$$

ここで、

$\sum a$；吸収量

$\sum s$；反射量

$\sum e$；鏡面反射量

$\sum i$；拡散反射量

$\sum tr$；透過量

$\sum t$；光線のエネルギーのトータル

である。

$\sum a$〜$\sum tr$ は、表面の物質、光線のエネルギー、面の法線と光源との角度θ、波長の関数である。ただし、実際の計算においては、計算時間、使用する記憶領域の制限等計算機資源を考慮してこれらの値の近似を行う必要がある。

物体からの光の反射量は、通常、その角度方向によって異なる。Fig.3.4にその例を示す。図中の矢印のような 45 度方向からの入射光の量は一定であっても、その反射光の角度方向による光量は差を生じる。また、この光量の差は塗装面の相違によって分布が異なり、たとえば、白くてつやのある塗装面は、反射角 45 度方向には鏡の正反射のような輝度を示すが、その他の反射角度では円形の均等拡散をする（図において、反射点から図形の輪郭までの直線距離がそれぞれの表面の反射光量を示す）。

今回作成しているプログラムでは、このような反射を Fig.3.5 のような鏡面反射成分と拡散反射成分で模擬している。鏡面反射成分については正反射方向に光線を 1 本発生し、拡散反射成分については乱数を用いて 1 本光線を発生する。したがって、体系中の各物体は、光の波長ごとにこの鏡面反射率と拡散反射率を有することになる。

以下に、各衝突モデル（反射、透過）に対するシミュレーションの概要を示す。

3.3.1 拡散反射による反射光成分

拡散反射においては、光があらゆる方向に均等に反射されていると考えるため、拡散反射された光線の強度変化は、視点の位置とは無関係に物体表面と入射光の角度で決定する。つまり、光線と面の法線の間の角度で衝突後の光線の強度が決定される。これは、Lambert の余弦則と呼ばれ [10]、これを用いて、拡散反射による強度変化を

$$E_{out} = E_{in} \times R_D \times \cos\theta \tag{3.3}$$

とする。

ここで、E_{out}, E_{in}は、それぞれ反射された光、入射光（光源からの光）の強度、R_Dは物体表面の拡散反射率（$= \sum i / \sum t$）である。また、Fig.3.6に示すように、θは面の法線と光源との角度である。光源として平行光線を用いた場合は、E_{in}を一定とする。これは、通常、平行光線として無限遠点からの光つまり太陽光を想定しているためである。光源として線光源あるいは点光源を用いた場合は、光のエネルギー（強度）が光源と衝突点との距離 r の二乗に比例して減衰するため、E_{in}に換えて、E_{in}/r^2を用いて光線の強度を計算する。

拡散反射後の光線の飛方向については、物体表面に相対する開空間に一様に分布するものとしてサンプリングする。

この場合の光線のウェイト w は、次のように変化させる。

$$w = w \times \frac{\sum i}{\sum t} \tag{3.4}$$

3.3.2 鏡面反射による反射光成分

鏡面反射によって物体表面において変化する光線の強度変化の計算には、Phong のモデルを用いる。Phong のモデルでは、鏡面反射による反射光の成分は、光線と入射光の面となす角度によって決まる [11]。つまり、鏡面反射による視点に対する寄与分を

$$E_{out} = E_{in} \times V(\theta) \times \tan^n \phi \tag{3.5}$$

で表す。

ここで、$V(\theta)$ は物体表面の鏡面反射率（$= \sum e / \sum t$）であり、たとえば高い鏡面反射率を用いれば、金属のような質感を表現することができる。E_{out}, E_{in}は、それぞれ反射された光、入射光（光源からの光）の強度である。拡散反射と同様、光源として平行光線を用いる場合は、E_{in}は一定、点光源を用いた場合は、E_{in}に換えて、E_{in}/r^2を用いる。また、Fig.3.6に示すように、ϕは視線と入射光のなす角度である。n は物体表面の滑らかさを表現する値で、n が大きくなるほど光沢のある面となる。

一般に画像中のハイライトと呼ばれる曲面に生じる帯状の白い部分は、鏡面反射によって作られるものである。ここで、$V(\theta)$ および n は、物質によって決定される関数または値であり、さまざまな質感はこれらの値を変化させることにより得られる。

光線のウェイト w は、次のように変化させる。

$$w = w \times \frac{\sum e}{\sum t} \tag{3.6}$$

3.3.3 透過

透明な物体を表示するため、透過性の物体表面に光線が衝突するまで追跡し、その後、与えられた屈折率にしたがって方向ベクトルが変化する次の光線をさらに追跡する。方向ベクトルは、Snellの法則と呼ばれる次の式によって変化させる [10]。

$$n_{out} \times \sin\theta_{out} = n_{in} \times \sin\theta_{in} \tag{3.7}$$

ここで、n_{out}, n_{in}は、それぞれ屈折前の物体の屈折率、屈折後の物体の屈折率である。

物体表面において屈折する光線のエネルギは、次の式によって変化する。

$$E_{out} = E_{in} \times \sum tr \tag{3.8}$$

光線のウェイトwは、次のように変化させる。

$$w = w \times \frac{\sum tr}{\sum t} \tag{3.9}$$

3.4 追跡する光線の次数

従来の光線追跡法では、追跡する光線の次数（反射の回数）に制限を設けるが、本システムにおいては、追跡する光線の次数には制限を設けていない。すなわち、追跡した光線が物体表面に衝突すると、さらに透過あるいは拡散／鏡面反射する光線を追跡する。

ただし、前述したように、追跡中の光線のウェイトが小さくなった（$w \leq w_{th}$）場合、ルシアン・ルーレット等を行い、追跡を継続するか否かを決定するとともに、ウェイトを調整する（生き残った場合、$w = w_{av}$）。

Table 3.1 Wavelength and color.

色	青	緑	赤
波長	480nm	540nm	680nm
群番号	1	2	3

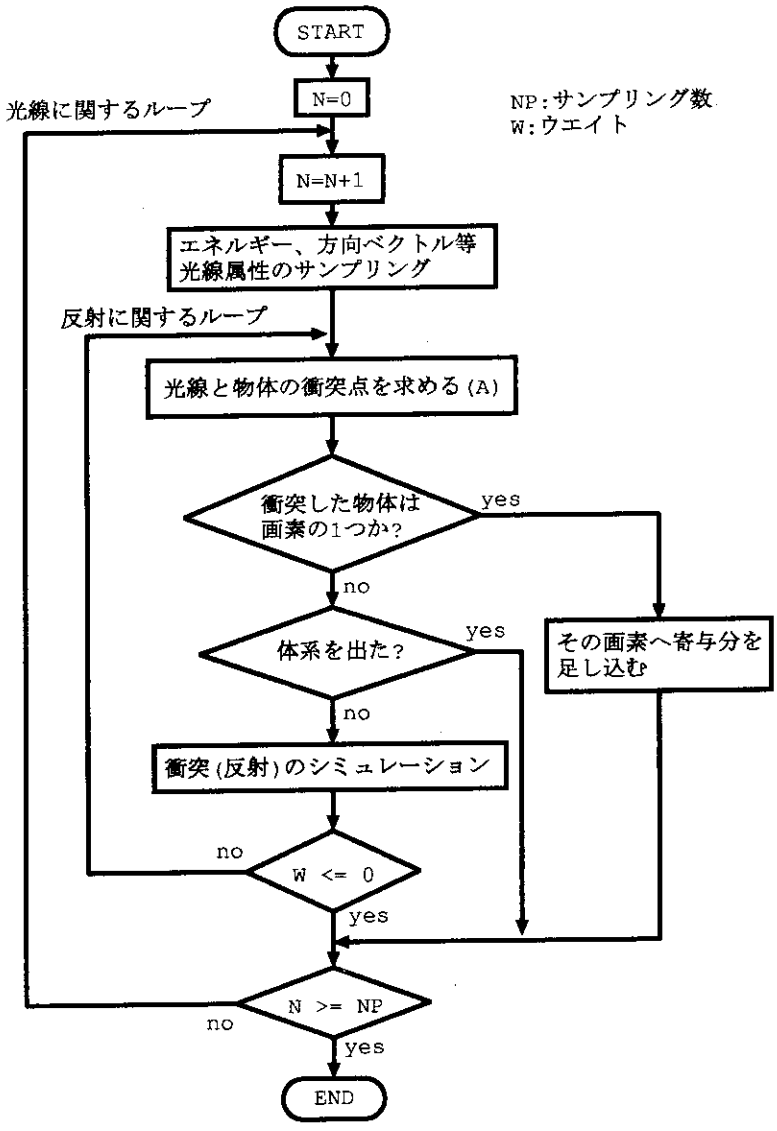

Fig. 3.1 Flow chart of RTMC(Ray Tracing by Monte Carlo method) code(1).

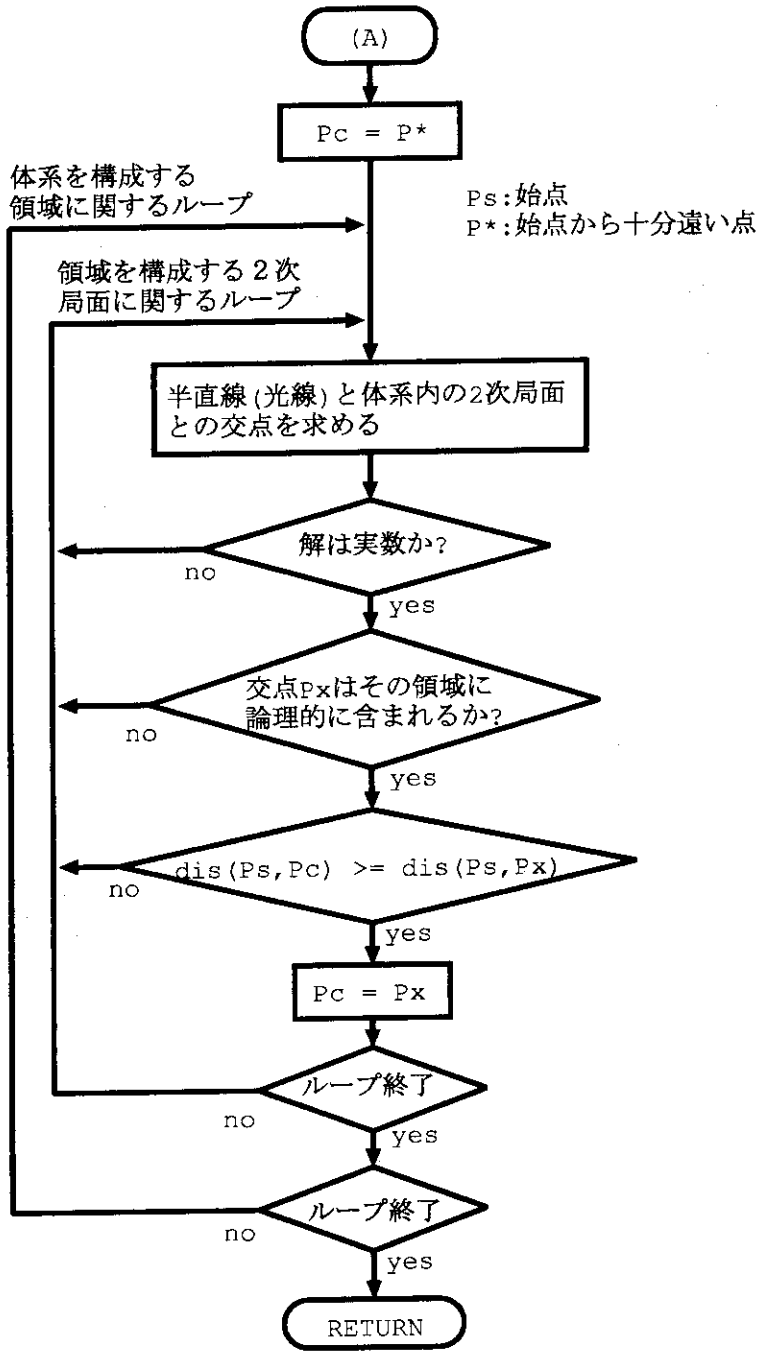

Fig. 3.2 Flow chart of RTMC code(2).

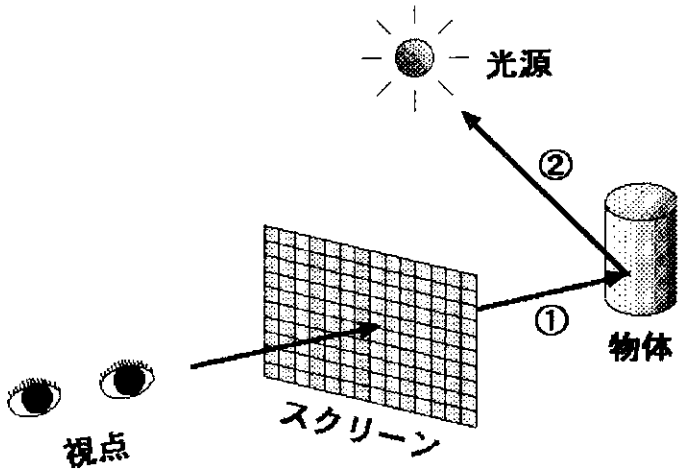

Fig. 3.3 Concept of conventional ray tracing.

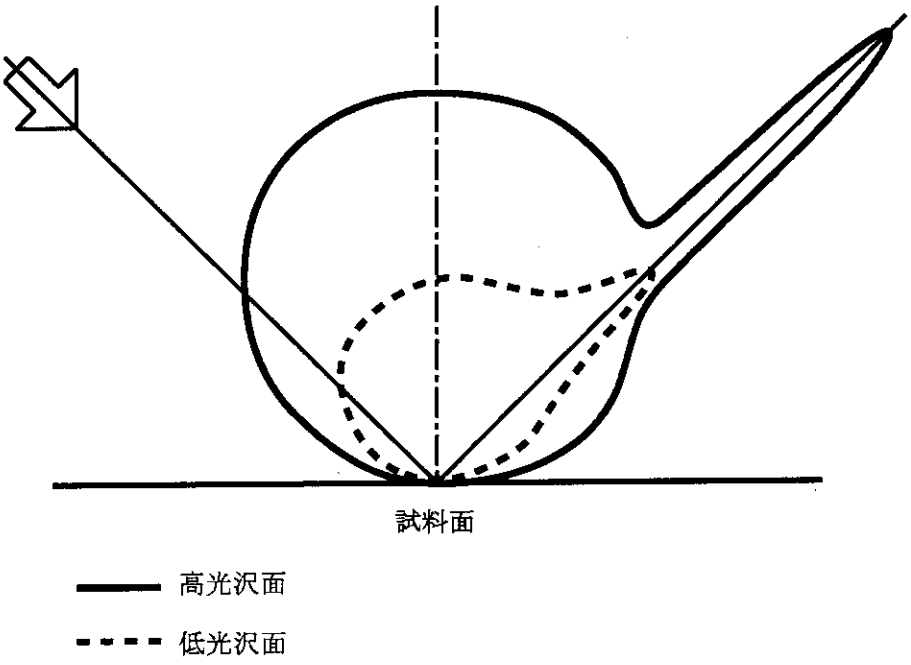

Fig. 3.4 An example of reflecting light distribution.

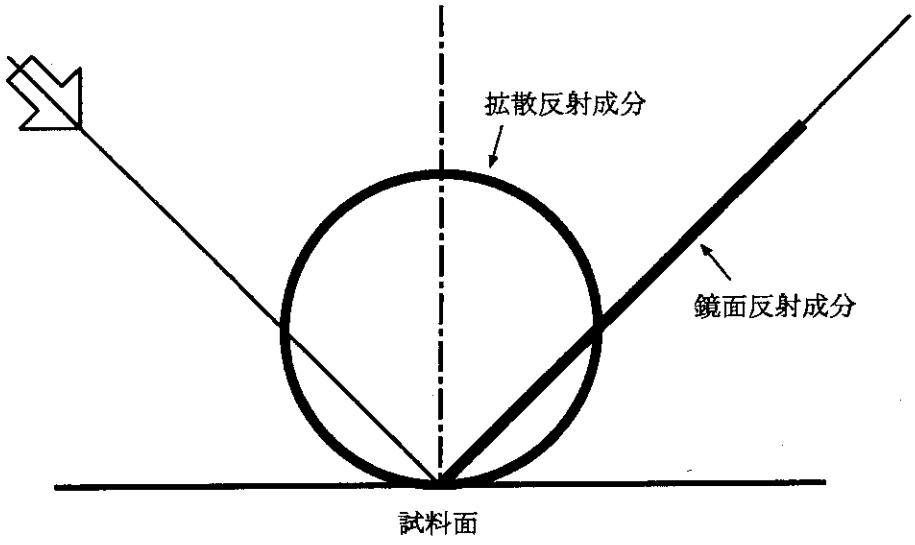

Fig. 3.5 Model of reflecting light distribution.

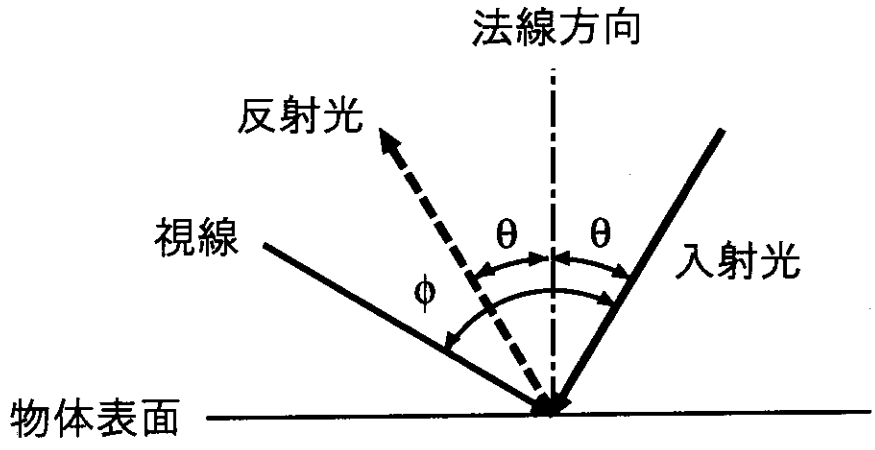

Fig. 3.6 Illustration of incident and reflection lights.

4. モンテカルロ法を適用した光線追跡による画像生成

4.1 形状データ

形状データの表現は、二次曲面の論理演算で3次元領域を表現する手法を用いた。

本研究においては、可視化の対象として、原子炉JRR-3M一次区画内の配管および主要な機器のモデル（二次曲面の枚数；約5,000、領域数；約1,000）を使用した。

形状データは、(1)機器・配管を構成する曲面を定義したサーフェイス・データ、(2)その曲面を組み合わせて機器・配管を構成する領域を定義したセル・データ、さらに、(3)各機器・配管表面の色すなわち光の波長ごとの反射率を定義した反射率データからなる。

サーフェイス・データは、以下のように定義されている。

- 1カラムから5カラムに、サーフェイス番号。
- 6カラムから10カラムに、サーフェイスの種類を表す記号。
- 11カラム以降に、サーフェイスを表現する係数（最大6個）を記述する。
- 係数が7個以上の場合は、次行に記述する。この場合は1カラムから10カラムを空白にして、11カラム以降に係数を記述する。

定義できるサーフェイスの種類および各サーフェイスを表現する係数をTable4.1およびTable4.2に示す。また、サーフェイス・データ、セル・データ、反射率データの簡単な例をそれぞれ、Table4.3、Table4.4およびTable4.5に示す。これらは、床面の上に1つの円筒形の立体が置かれた環境の例である。

Table4.4に示したセル・データは、Table4.3で定義したサーフェイスの論理演算により領域（立体）を定義している。すなわち、領域1は、サーフェス・データの1番、2番の平面の上下と3番の無限円筒（内部）との論理積で円筒立体を定義している。同様に領域2は、6枚の平面による直方体を2つ用いて、有限空間内の床を定義している。

Table4.5では、セル・データにより定義した領域の色（光の波長毎の反射率）を定義している。1行目（m1）は、セル・データの領域1すなわち円筒の反射率を定義し、2行目（m2）は床面のそれを定義している。各行の数値は、最初の3つがそれぞれ、680nm、540nm、480nmの波長の光の鏡面反射率、次の3つが同様に3波長の光の拡散反射率、次の3つが同様の3波長の透過率、次の1つは透過の際の屈折率、最後は鏡面反射のべき乗係数である。

4.2 入力モデル

今回作成した画像生成プログラムのための主な入力データをTable4.6に、パラメータをTable4.7にそれぞれ示す。

Table4.6中の光源（so）については、その種類として、点光源、線光源、平行光線の3種が指定できる。点光源を指定した場合、光源の位置及び強度を入力する。平行光線を指定した場合は、光源位置の代わりに光線の方向を入力する。また、光源は複数個の指定が可能である。物体表面の光学的特性については、Mカードを用いて定義する。例えば、鏡面反射率は、各RGBに対し、0.0～1.0の範囲で指定する。拡散反射率、透過率についても、各RGBに対し、0.0～1.0の範囲で指定する。その他、屈折率、鏡面反射べき乗係数を指定する。

4.3 プログラム構成

プログラム構成の概略（主たるサブルーチン）をFig.4.1に示す。プログラムの規模は、約4,400ステップ（パラメータ文等を含めて約4,900ステップ）、サブルーチン数は68個である。

サブルーチンPRIPRC以下のルーチン群は、前処理を行なう。PRIPRCの下位ルーチンINPUT以下は入力データの読み込みを行い、MKSCRN以下はスクリーンの設定を行う。

サブルーチンRAY以下のルーチン群は、光線追跡を行なう。サブルーチンRAYの下位ルーチンSOURCE以下は、光線の生成、CROSS2以下は光線と領域の交差テスト、ANGLE以下は反射角等の計算、TRACE以下は遮蔽物の有無のチェック、COLSET以下は寄与エネルギーの計算をそれぞれ行う。

サブルーチンPREPRC以下のルーチン群は、後処理を行なう。サブルーチンPREPRCの下位ルーチンRGBSETは、寄与エネルギー値とRGB値の変換、RGBOUTは画像の出力を行う。

また、Fig.4.1に示していないが、モンテカルロ・シミュレーションに不可欠の乱数生成処理を行なうルーチン群として、SEEDNおよびRANU21がある。前者は、MAINから呼ばれ、乱数の種(シード)を生成する。後者は、SOURCEの下位ルーチンであり、SEEDNで生成したシードを用いて乱数を生成する。

なお、プログラム構成および引数等の詳細については付録2に示す。

4.4 高速化手法

4.4.1 バウンダリ・ボリュームの導入

バウンダリ・ボリュームは、光線追跡法で画像を生成する際、光線と環境中の物体との交差テストの計算コストを低減するために用いられる[13]。

光線追跡法では、光線がどの物体に当たるかを知るために、物体と光線の交差テストを行う必要がある。交差した物体のうち、最も光線の出発点に近いものが、光線の当たる物体である。光線の出発点と方向は任意であるから、環境中のすべての物体が交差テストの対象になる。

ある物体に対し交差テストを行う際に必要な時間は、その物体の幾何学的表現に依存する。一般的には多面体の交差テストはその面の数に依存するが、他の形状、たとえば回転体の表面やパラメトリック曲面に対しては、さらに多くの時間が必要になる。原子力施設のような

大規模で複雑な環境においては、環境の形状は極めて多数の多面体の組み合わせによって表現される。

このため、複雑な物体を球や平行六面体のような簡単な多面体でそれを包み込んでしまうことが望ましい場合が多い。このような手法をバウンダリ・ボリュームと呼ぶ。すなわち、光線がバウンダリ・ボリュームと交差するときにのみバウンダリ・ボリュームで包み込まれた複雑な物体と光線との交差テストを行い、光線がバウンダリ・ボリュームと交差しない場合には内部の複雑な物体と光線との交差テストを行う必要はない。したがって、バウンダリ・ボリュームが大きくとられれば、この手法は光線と物体との交差テストに対して非常に計算コストを軽減する効果をもつ。

本研究では、バウンダリ・ボリュームとして、球を採用し、前述の JRR-3M 一次区画内の機器について、機器毎に１つの球で包み込む方式を導入している。バウンダリ・ボリュームとして球を採用した理由は、球が最も少ない数（１つ）の曲面で表現される多面体であるためである。

Fig.4.2に三次元形状をバウンダリ・ボリュームで包み込んだ例を示す。この場合、光源から発せられた光線（点線）はバウンダリ・ボリューム（球 1～3）と交差テストを行い、光線と交差するバウンダリ・ボリューム（球 2）が包含している形状についてのみ各曲面と光線との交差テストを行う。

4.4.2　インポータンス・サンプリングの適用

インポータンス・サンプリングとは、ルシアン・ルーレットと同様粒子輸送モンテカルロ・コードで用いられている分散低減法である。粒子の領域分割を行うため、各領域にインポータンスを設定し、他の領域と比較して体積の小さな領域に対し大きな値を設定することによってサンプリング粒子数に依存して体積の小さな領域が無視される（サンプリング粒子数が少ない場合、その領域に進入する粒子が極端に少ないためにその領域の影響が結果に反映されない）ことを防止する。具体的には、インポータンスの大きな領域に粒子が進入した場合、進入した粒子を複数に水増し（分割といってもいい）することによって領域の影響が結果に伝播するようにする。ただし、このとき粒子のウェイトの総和は保存する（１つの粒子を２つにした場合、２つの粒子のウェイトはそれぞれもとの粒子のウェイトの半分に設定する）ことにより、全体的なバランスを保持する。

インポータンスの概念の本研究への適用を考慮すると、体積の小さな物体、正確には面積の小さな曲面に入射する光線数を水増しすることにより、その曲面の像の取りこぼしを防止することに相当するが、そのためには曲面に入射する光線を検出しこれを加工する必要がある。ところが、光線の出発点と方向は任意であるから、どの光線が該当曲面に入射するかは検出不能である。したがって、このままではインポータンスの手法は適用できない。

しかしながら、インポータンスの高い領域での反射光線数を水増しすることを考えると、インポータンスの概念を導入することは可能である。順方向のレイトレーシングでは、物体の表面における鏡面反射光は、光線の波長、入射角および物体表面の反射率特性により、そのベクトルが決定する。したがって、この鏡面反射光のベクトルは、インポータンスとは全

く独立のものである。ところが、拡散反射光に関してはモンテカルロ・コード同様乱数を用いてシミュレーションを行っており、インポータンスの導入が有効となる。すなわち、インポータンスの高い領域で反射した場合の拡散反射光は複数に分割し、シミュレーションを行う。もちろん、この場合も光線のウェイトの総和は保存することにより全体のバランスを保持する。粒子輸送モンテカルロ・コードでは、インポータンスの高い領域に進入した際に粒子の分割を行うが、本プログラムでは、インポータンスの高い領域で衝突（反射）した際に光線の分割を行う。

4.4.3 ベクトル処理による高速化

本コードは光線の光源における発生から消滅するまでを追跡するのであるが、各光線の間の相互作用を考慮していないので、各光線の挙動は互いに独立である。従って、複数の光線を同時に追跡する、すなわちベクトル処理が可能である。

個々の光線は独立であるが、処理（イベントと呼ぶ）内容は個々に異なる。イベントごとに光線を分類して、イベント単位に処理を行う。これをイベントバンク方式という。例えば、光線と領域との交差判定をした場合、領域と交差した光線、領域と交差しなかった光線に分類され、イベントごとにベクトル処理が行われる。

各光線の属性データは間接番地で参照される。間接番地参照は、連続番地参照と異なり、主記憶からベクトル・レジスタへの転送速度が遅く、このため、演算パイプラインに遊びが生じ、ベクトル・プロセッサの性能が低下する。今回計算に使用した高速モンテカルロ装置[14]は間接番地参照の転送を行うロード／ストア・パイプラインを強化しているため、間接番地データは直接番地データ同様、高速に転送される。

また、メモリ容量には限界があるため、追跡する全光線（＝ヒストリ数）を同時に処理することはできない。まず、同時に追跡する光線数（ベクトル処理する光線数）を適当に決めて処理する。光線の消滅によって現在処理している光線数というのは減少してくる。そこで光線の消滅によって生じた空領域に新たに追跡を開始する光線を補充追加する。これを繰り返すことで全ヒストリが追跡されるという方法を採用している。

4.4.4 並列処理による高速化

並列処理には、マクロタスク機能を用いる方法とマイクロタスクを用いる方法がある。前者がサブルーチンのような大きな単位の並列性に着目して並列化するのに対し、後者はDOループや文の集まりのような比較的小さな並列性に着目して並列化する。

光線追跡法の大きな特徴は、追跡する光線間に依存関係がないため、各光線をまったく独立に扱うことができることであり、本コードもきわめて良好な並列性を有している。本コードは、光線の発生から消滅までの一連の処理を並列に行うことが可能であることから、細かい並列性には注目せず、マクロタスク機能を用いて並列化された。

並列処理するには、まず、主タスクから並列処理部分を子タスクとして生成する。すなわち、主タスクから並列処理部分であるサブルーチンRAYを子タスクとして生成する。子タスクは並列処理関数PTFORKで呼び出した処理の正確なコピーである。ヒストリは負荷分散が

均等になるように各タスクに分割（粒子分割）する。子タスクの消去は並列処理関数PTJOINを用いて行われ、待ち合わせ制御がされる。すべての子タスクの消去が確認されると、次の処理が開始される。

また、並列処理の高速な処理を疎外する原因となる排他処理を必要としないコーディングを行うように注意した。まず、各タスクで独立に制御できるような乱数を使用した。また、統計処理は並列処理部分では各タスク用に用意された配列に統計値を積算していき、すべてのタスクが終了してから各画素に対する寄与分を集計して全体の統計値を求めるようにした。

マクロタスク機能により並列化されたメインルーチンをFig.4.3に示す。

Table 4.1 Definition format of surfaces.

面種類	記号	条件	数式	係数入力並び
平面	P	任意	$Ax+By+Cz-D=0$	A,B,C,D
	PX	yz 平面に平行	$x-D=0$	D
	PY	xz 平面に平行	$y-D=0$	D
	PZ	xy 平面に平行	$z-D=0$	D
球面	SO	中心が原点	$x^2+y^2+z^2-R^2=0$	R
	S	任意	$(x-X)^2+(y-Y)^2+(z-Z)^2-R^2=0$	X,Y,Z,R
	SX	中心が x 軸上	$(x-X)^2+y^2+z^2-R^2=0$	X,R
	SY	中心が y 軸上	$x^2+(y-Y)^2+z^2-R^2=0$	Y,R
	SZ	中心が z 軸上	$x^2+y^2+(z-Z)^2-R^2=0$	Z,R
円柱面	C/X	中心軸が x 軸に平行	$(y-Y)^2+(z-Z)^2-R^2=0$	Y,Z,R
	C/Y	中心軸が y 軸に平行	$(x-X)^2+(z-Z)^2-R^2=0$	X,Z,R
	C/Z	中心軸が z 軸に平行	$(x-X)^2+(y-Y)^2-R^2=0$	X,y,R
	CX	中心軸が x 軸上	$y^2+z^2-R^2=0$	R
	CY	中心軸が y 軸上	$x^2+z^2-R^2=0$	R
	CZ	中心軸が z 軸上	$x^2+y^2-R^2=0$	R
円錐面	K/X	中心軸が x 軸に平行	$\sqrt{(y-Y)^2+(z-Z)^2}-T(x-X)=0$	$X,Y,Z,T^2,\pm1^\dagger$
	K/Y	中心軸が y 軸に平行	$\sqrt{(x-X)^2+(z-Z)^2}-T(y-Y)=0$	$X,Y,Z,T^2,\pm1^\dagger$
	K/Z	中心軸が z 軸に平行	$\sqrt{(x-X)^2+(y-Y)^2}-T(z-Z)=0$	$X,Y,Z,T^2,\pm1^\dagger$
	KX	中心軸が x 軸上	$\sqrt{y^2+z^2}-T(x-X)=0$	$X,T^2,\pm1^\dagger$
	KY	中心軸が y 軸上	$\sqrt{x^2+z^2}-T(y-Y)=0$	$Y,T^2,\pm1^\dagger$
	KZ	中心軸が z 軸上	$\sqrt{y^2+z^2}-T(z-Z)=0$	$Z,T^2,\pm1^\dagger$
楕円面 双曲面 放物面	SQ	各軸が x,y,z 軸に平行でない	$A(x-X)^2+B(y-Y)^2+C(z-Z)^2$ $+2D(x-X)+2E(y-Y)+2F(z-Z)$ $+G=0$	A,B,C,D,E,F,G,X,Y,Z
円柱面 円錐面 楕円面 双曲面 放物面	GQ	各軸が x,y,z 軸に平行でない	$Ax^2+By^2+Cz^2$ $+Dxy+Eyz+Fzx$ $+Gx+Hy+Iz+J=0$	A,B,C,D,E,F,G,H,I,J

† ± 1 only use for 1 sheet cone.

Table 4.2 Definition format of surfaces (continuation).

面種類	記号	条件	数式	係数入力並び
輪環面（楕円断面、円断面）	TX	ZY 平面に平行	$(x-X)^2/B^2$ $+(\sqrt{(y-Y)^2+(z-Z)^2}-A)^2/C^2-1=0$	X,Y,Z,A,B,C
	TY	XZ 平面に平行	$(y-Y)^2/B^2$ $+(\sqrt{(x-X)^2+(z-Z)^2}-A)^2/C^2-1=0$	X,Y,Z,A,B,C
	TZ	XY 平面に平行	$(z-Z)^2/B^2$ $+(\sqrt{(x-X)^2+(y-Y)^2}-A)^2/C^2-1=0$	X,Y,Z,A,B,C
	TG	任意	$Ax^4+By^4+Cz^4$ $+Dx^3y+Exy^3+Fy^3z+Gyz^3+Hz^3x+Izx^3$ $+Jx^2y^2+Ky^2z^2+Lz^2x^2$ $+Mx^2yz+Nxy^2z+Oxyz^2$ $+Px^3+Qy^3+Rz^3$ $+Sx^2y+Txy^2+Uy^2z+Vyz^2+Wz^2x+AAzx^2$ $+BBxyz$ $+CCx^2+DDy^2+EEz^2$ $+FFxy+GGyz+HHzx$ $+IIx+JJy+KKz+LL=0$	$A,B,C,D,E,F,G,H,I,$ $J,K,L,M,N,O,P,Q,R,$ $S,T,U,V,W,AA,BB,CC,$ $DD,EE,FF,GG,HH,II,$ JJ,KK,LL

Table 4.3 An example of surface data.

```
1    P +0.000000000000E+00 +0.000000000000E+00 +1.000000000000E+00 +0.000000000000E+00
2    P +0.000000000000E+00 +0.000000000000E+00 -1.000000000000E+00 -7.000000000000E+01
3    GQ-6.250000000000E-04 -6.250000000000E-04 +0.000000000000E+00 +0.000000000000E+00
       +0.000000000000E+00 +0.000000000000E+00 +0.000000000000E+00 +0.000000000000E+00
       +0.000000000000E+00 +1.000000000000E+00
11   P +0.000000000000E+00 +0.000000000000E+00 +1.000000000000E+00 +0.000000000000E+00
12   P +0.000000000000E+00 +0.000000000000E+00 +1.000000000000E+00 -1.000000000000E-02
13   P +0.000000000000E+00 +0.000000000000E+00 +1.000000000000E+00 +2.001000000000E+05
14   P +0.000000000000E+00 +0.000000000000E+00 +1.000000000000E+00 +2.000000000000E+05
21   P +0.000000000000E+00 +1.000000000000E+00 +0.000000000000E+00 -1.001000000000E+05
22   P +0.000000000000E+00 +1.000000000000E+00 +0.000000000000E+00 -1.000000000000E+05
23   P +0.000000000000E+00 +1.000000000000E+00 +0.000000000000E+00 +1.001000000000E+05
24   P +0.000000000000E+00 +1.000000000000E+00 +0.000000000000E+00 +1.000000000000E+05
31   P +1.000000000000E+00 +0.000000000000E+00 +0.000000000000E+00 -1.001000000000E+05
32   P +1.000000000000E+00 +0.000000000000E+00 +0.000000000000E+00 -1.000000000000E+05
33   P +1.000000000000E+00 +0.000000000000E+00 +0.000000000000E+00 +1.001000000000E+05
34   P +1.000000000000E+00 +0.000000000000E+00 +0.000000000000E+00 +1.000000000000E+05
```

Table 4.4 An example of cell data.

```
1  1     1 2 3
2  2     12 -13 21 -23 31 -33 #(-11 14 -22 24 -32 34)
```

Table 4.5 An example of reflection data.

```
 m1      0.3 0.15 0.1   0.6 0.3 0.2   0. 0. 0.    1.   3
 m2      0.1 0.4 0.3   0.05 0.2 0.15   0. 0. 0.    1.   1
```

Table 4.6 Input data for RTMC code.

データ名	意味	データ内容
VIEW	視点	座標 (x, y, z)
SCRN	スクリーン	中心点座標、画素数
SCAL	画像のスケール係数	実数
SO	光源	光源種類、光源位置座標、光線のエネルギー
M	セル属性	セル表面の鏡面反射率、散乱反射率等
HIST	追跡する光線数	整数

Table 4.7 Parameters for RTMC code.

パラメータ名	意味
INSURF	領域チェックの際に使用する WORK 配列の大きさ
MLN	スクリーンの横の大きさ
IYYSIZ	スクリーンの縦の大きさ
NPART	ベクトル処理粒子数
maxc	1セルに使用できる最大項目数
maxs	最大面数
maxnc	最大セル数
maxso	最大光源数
iic	セル・カード入力機番
ioc	セル・カード出力機番
iis	サーフェイス・カード入力機番
ios	サーフェイス・カード出力機番
iid	データ・カード入力機番
ioe	データ・カード出力機番
nkd	カードの種類の最大数
maxbv	バウンダリ・ボリュームの最大数
nrr	1回に発生させる乱数の量
maxcpu	最大タスク数
cdiv	寄与エネルギーの離散化係数
dnorm	光の減衰係数
wlow	ルシアン・ルーレットを行うしきい値
wavg	ルシアン・ルーレットに生き残った場合のウェイト

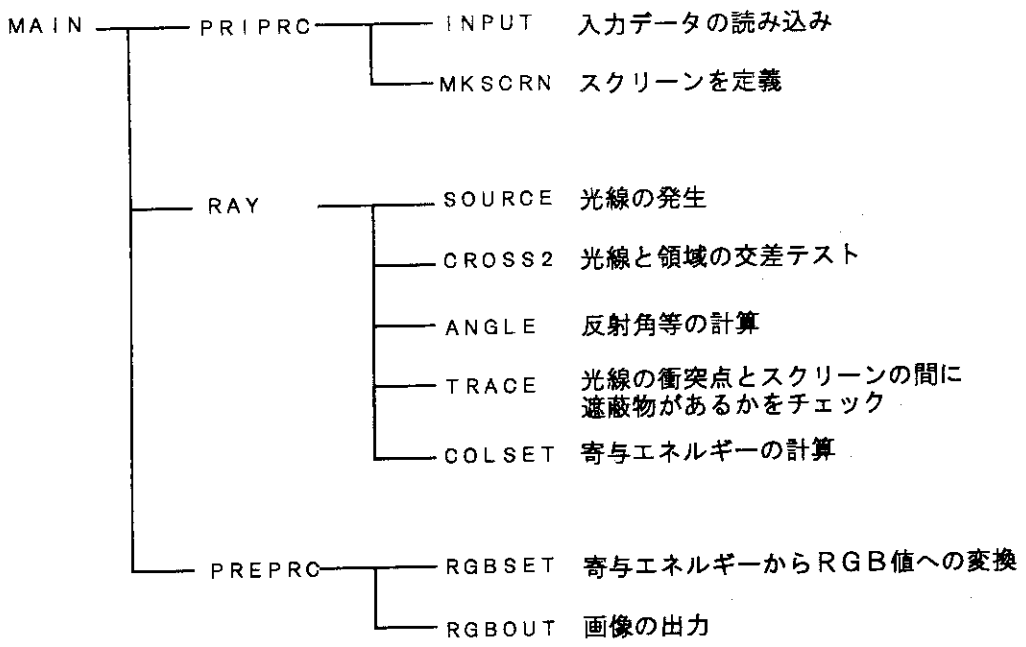

Fig. 4.1 Program structure.

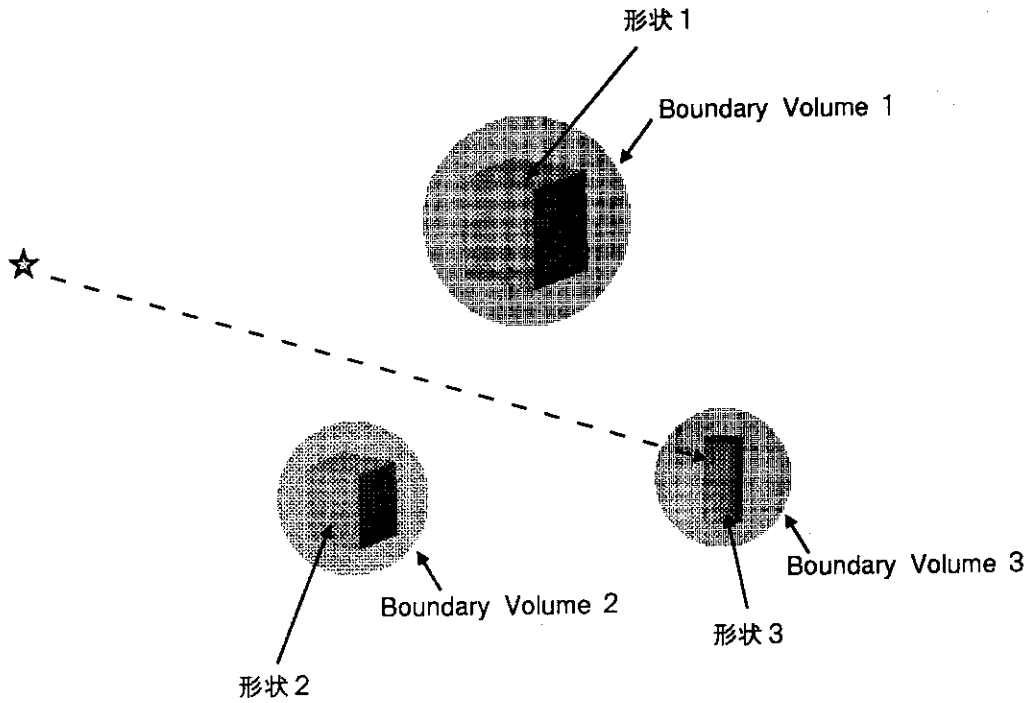

Fig. 4.2 Effect of boundary volume.

```
      PROGRAM MAIN
c
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
c
      external ray
      include(PARAM)
      include(DCARD)
      include(COLOR)
      dimension rrp(nrr,maxcpu),ixr(maxcpu),locrp(maxcpu)
      dimension nnpsp(maxcpu)
      integer*4 tvar(3),tparm(3)
      data tparm/0,0,0/
      data tvar /0,0,0/
c
      IX0=0                          --+
      CALL SEEDN(IX0,IXR,MAXCPU)      | 乱数の種と
      CALL RANU21                     | 係数等を
      DO 10 I=1,MAXCPU                | タスク分
         LOCRP(I) = NRR               | 用意する
   10 CONTINUE                       --+
c
      CALL PRIPRC
c
      nnn1=nnps/maxcpu               --+
      nnn2=nnps-(maxcpu-1)*nnn1       |
      if(maxcpu.ge.2)then             |
         do 20 I=1,maxcpu-1           | 粒子分割
            nnpsp(i)=nnn1             |
   20    continue                     |
      endif                           |
      nnpsp(maxcpu)=nnn2             --+
c
      call ptfork(tvar(3),tparm(3),ray,                                --+
     +         COLR(1,1,4),COLG(1,1,4),COLB(1,1,4),                     |
     +         COLR2(1,1,4),COLG2(1,1,4),COLB2(1,1,4),ss(1,1,4),        |
     +         ncolor(1,1,4),ixr(4),rrp(1,4),locrp(4),nnpsp(4))         |
      call ptfork(tvar(2),tparm(2),ray,                                | 子
     +         COLR(1,1,3),COLG(1,1,3),COLB(1,1,3),                     | タ
     +         COLR2(1,1,3),COLG2(1,1,3),COLB2(1,1,3),ss(1,1,3),        | ス
     +         ncolor(1,1,3),ixr(3),rrp(1,3),locrp(3),nnpsp(3))         | ク
      call ptfork(tvar(1),tparm(1),ray,                                |
     +         COLR(1,1,2),COLG(1,1,2),COLB(1,1,2),                     |
     +         COLR2(1,1,2),COLG2(1,1,2),COLB2(1,1,2),ss(1,1,2),        |
     +         ncolor(1,1,2),ixr(2),rrp(1,2),locrp(2),nnpsp(2))        --+
c
      CALL RAY(COLR(1,1,1),COLG(1,1,1),COLB(1,1,1),                    --+ 主
     +         COLR2(1,1,1),COLG2(1,1,1),COLB2(1,1,1),ss(1,1,1),        | タ
     +         ncolor(1,1,1),ixr(1),rrp(1,1),locrp(1),nnpsp(1))        --+ ス
c                                                                          ク
      call ptjoin(tvar(3))           --+
      call ptjoin(tvar(2))             | join処理
      call ptjoin(tvar(1))           --+
c
      CALL PREPRC
c
      STOP
      END
```

Fig. 4.3 Parallelization of RTMC code.

5. 結果および考察

5.1 逆光線追跡法との比較

従来の光線追跡法、すなわち、逆光線追跡法 (視線追跡法) との比較を述べる。これまで述べてきたように、本手法、すなわち、順方向光線追跡法の利点は、光源の厳密な幾何形状表現が可能な点にある。例えば、照度分布を持った線／面光源の下で観測される「ぼんやりとした影」の描写が可能である。本手法と従来手法、すなわち、逆光線追跡法との比較を Fig.5.1及び Fig.5.2に示す。Fig.5.1が従来手法、Fig.5.2が本手法による画像である。可視化には、JRR-3M 一次区画内の PCS 主ポンプを用いた。Fig.5.2では、光源として 25 個の点光源を用いて面光源を近似した。1 光源あたり、5 千万本、合計 12 億 5 千万本の光線を追跡している。Fig.5.1においては、影領域がはっきりしているのに対し、Fig.5.2においては、面光源の効果により、床面にぼんやりとした影ができていることがわかる。また、物体表面の光沢についても、Fig.5.1の場合、一様であるのに対し、Fig.5.2の場合、自然な反射が表現されている。

しかしながら、計算時間については、従来手法と比較し、大きな欠点がある。すなわち、Fig.5.1の生成時間は、Monte-4 上で約 500 秒であるのに対し、Fig.5.2の生成には、その 1,000 倍程度の計算時間を要する。本手法の場合、画像生成時間は、光源の数、追跡する光線の数に依存する。追跡する光線の数が大きければ大きいほど、キメの細かな画像が得られる。計算時間を決定する画像パラメータが違うため (「同程度の画質」の定義が難しいため)、両者の比較は単純ではないが、本手法による画像生成には、次節で述べるような高速化手法を用いても、従来手法と比較し、100 ～ 1,000 倍程度の計算時間を要する。

5.2 計算時間に関する考察

各種の計算時間短縮化手法の効果について評価するため、Table5.1に示す種々の条件で計算を実施し、その処理時間を測定した。測定に用いた環境 (可視化したモデル) は、前節で示した JRR-3M の 1 次冷却系区画であり、画像生成においては、100 万本の光線を追跡した。ただし、表中の*印 (推定値) については、処理時間がきわめて大きいことから 10 万ヒストリの光線追跡にとどめて測定を行い、得られた測定値を 10 倍した値を記載している。表からわかるように、バウンダリ・ボリュームやルシアンルーレットによるソフト面からのアプローチに加え、モンテカルロ・パイプラインやベクトル並列処理機能を有する高速モンテカルロ装置によるハード面からのアプローチにより処理の高速化を図り、スカラ処理と比較して、700 倍程度の速度向上率を得た。高速化の詳細については、次節以降に述べる。

また、Table5.2に RTMC コードの各サブルーチンごとの計算コストの解析結果を示す。最もコストの高いサブルーチン LGEVAL はサブルーチン CHKCEL より呼び出されており、この両者すなわち LGEVAL と CHKCEL で全体の計算時間の約 84 %を占めていることがわ

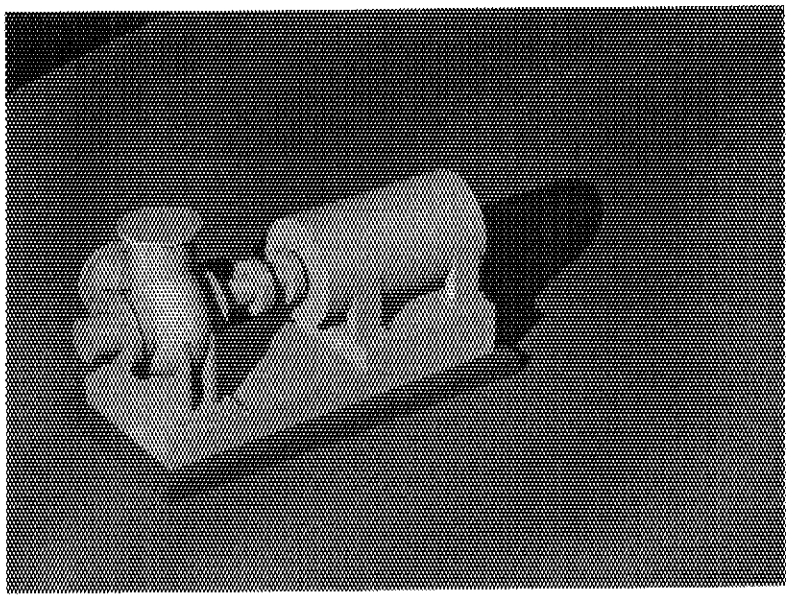

Fig. 5.1 An example of three dimensional image by conventional ray tracing algorithm.

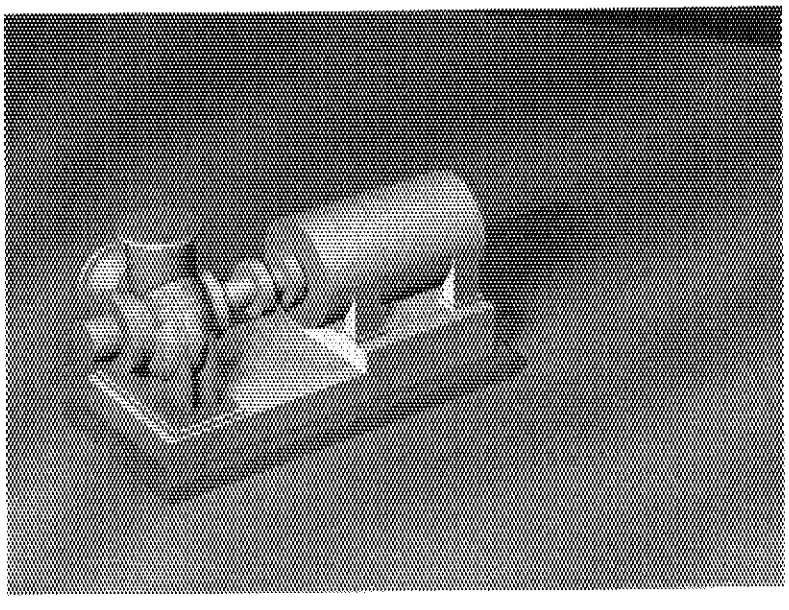

Fig. 5.2 An example of three dimensional image by rtmc code.

かる。

5.2.1 分散低減法による効果

分散低減法の効果は、ルシアン・ルーレットにかけるウェイト値のしきい値 w_{th} を変化させた場合の計算処理時間の変化によって評価できる。

Table5.1中の6番に示す結果（†印）は、同表中の8番と同一条件下でしきい値 w_{th}のみを1/10にした場合の計算処理時間を表したものである。すなわち、分散低減法の適用を（ウェイト値が8番のウェイト値の1/10になるまで）遅らせると、処理時間はおよそ2倍弱になる。本来、分散低減法の適用を全く行わない場合と比較すべきであるが、その場合の計算処理時間はきわめて大きくなるため、処理時間測定の都合から上記のような推定値を用いた。

5.2.2 バウンダリ・ボリュームによる計算時間の短縮

バウンダリ・ボリュームによる計算時間短縮の効果については、その原理から概算することができる。本来、ある形状を構成するすべての曲面に対し光線との交差テストを行うべきところ、その形状を包含する1つの球面でその交差テストを代替し、球面と光線が交差していない場合には曲面との交差テストを行わないことから、例えば20枚の曲面から構成される形状を1つのバウンダリ・ボリューム（球）で包んだ場合、この球面と光線が交差していなければ、この形状に対する交差テストは約20倍高速化されることになる。したがって、1つのバウンダリ・ボリュームに包含される曲面の枚数によって計算時間の短縮率が概算できる。ただし、曲面の種類によって交差テストに要する時間が大きく異なること、1つの光線が複数のバウンダリ・ボリュームと交差する可能性があること等から概算した値はあくまでも目安ということになる。

本報告で画像生成を行ったJRR-3Mのデータを参照すると、その環境は約3,800枚の曲面により構成されている。一方、今回設定したバウンダリ・ボリュームはTable5.3に示したとおりである。したがって、どのバウンダリ・ボリュームとも交差しない光線については、バウンダリ・ボリュームの導入によって交差テストが約220倍（3800/17）高速化され、1つのバウンダリ・ボリュームと交差する光線については約17倍、n個のバウンダリ・ボリュームと交差する光線については約$17/n$倍の高速化が見込める。

Table5.1に示した5番のデータと8番のデータを比較すると、バウンダリ・ボリュームの導入による計算処理時間の高速化率は約17倍である。したがって、画像全体として1本の光線がほぼ1個のバウンダリ・ボリュームと交差している状態が推察でき、所期の目標に近い高速化率を本ソフトウェアが実現できたことを示す。

これに対し、Table5.1中の2番と3番のデータを比較すると、計算処理時間の高速化率は30倍を越えている。これは、2番のデータを測定する際、追跡する光線数を減少させたために(注；2番の処理時間は推定値)、どのバウンダリ・ボリュームとも交差しない光線の割合が増加したことを起因している。言い換えれば、画像解像度に対して安定した画像を提供し得る光線数よりも追跡光線数が少な過ぎることを示している。

5.2.3 高速モンテカルロ装置によるベクトル並列処理

本ソフトウェアでは、二次あるいは四次曲面の論理演算によって表現された三次元領域において、光線を追跡する際、ある点がどの領域に含まれるかという判定を行う処理が不可欠である。この処理のために二値論理式の評価が必要となり、多くの時間をこの処理に費やしている。Table5.2に示したとおり、この二値論理式の評価を行うルーチン（LGEVAL）が最も高い計算コストを占めている。

しかし、高速モンテカルロ装置では、その特長であるモンテカルロ・パイプラインの1つ、領域判定パイプラインを適用することによって、この二値論理式の評価を高速化することができる（付録1参照）。

Table5.1を参照すると高速化率は概ね2倍であるが、領域判定パイプラインの効果を確認するためには、もっと詳細に計算実行の様子を観察する必要がある。高速モンテカルロ装置に装備されている動的アナライザを用いて、サブルーチン CHKCEL に要した CPU 時間を計測すると、Table5.1と同一の条件下で 4,722[sec] であった。このうち、LGEVAL に要した（LGEVAL は CHKCEL から呼び出されている）CPU 時間は、2,730[sec] であった。一方、二値論理式の評価を領域判定パイプライン、すなわちハードウェアを用いて実行すると（当然サブルーチン LGEVAL は不要になる）、サブルーチン CHKCEL に要する CPU 時間は 2,013[sec] に短縮された。この両者の差異が、すなわち LGEVAL に要した CPU 時間の短縮分であるから

$$\text{高速化率 [\%]} = \frac{CPU\text{ 時間短縮分}}{\text{パイプライン適用前の } CPU\text{ 時間}} = \frac{4,722 - 2,013}{2,730} \simeq 0.992$$

倍率に直すと、1/(1 - 0.992) であるから、領域判定パイプラインは LGEVAL に対して、およそ 125 倍の速さで二値論理式を評価していることがわかる。

また、ベクトル処理による高速化率については、Table5.1の4番と8番を比較すると、67.9 [%] の CPU 時間短縮、倍率（速度向上率）にしておよそ 3.12 倍の高速化を達成している。前述のアナライザによる計測の結果、ソフトウェア全体のベクトル化率は 81.65[%]、平均ベクトル長は 31 である。高速モンテカルロ装置のベクトルレジスタ長は 64 で、通常のスーパーコンピュータに比較してやや短い。時としてこれはベクトル演算において短所となることがあるが、今回のソフトウェアに関しては上記の結果から関係ない。

並列処理（4CPU）による高速化率については、Table5.1の8番と9番を比較して、66.2[%] の CPU 時間短縮、すなわち、約 2.96 倍の速度向上率を達成している。

Table 5.1 Computational time of the code.

番号	計算条件					処理時間 (sec)
	スカラ／ベクトル	モンテカルロパイプライン	分散低減法	バウンダリボリューム	プロセッサ台数	
1	スカラ	—	—	—	1	554,910*
2	スカラ	—	適用	—	1	323,220*
3	スカラ	—	適用	適用	1	9,756
4	スカラ	適用	適用	適用	1	7,231
5	ベクトル	適用	適用	—	1	39,651
6	ベクトル	適用	$w_{th}/10$	適用	1	4,489†
7	ベクトル	—	適用	適用	1	4,939
8	ベクトル	適用	適用	適用	1	2,321
9	ベクトル並列	適用	適用	適用	4	785

*は推定値。

$w_{th}/10$ は、通常の 1/10 の閾値による分散低減法適用を示す。

Table 5.2 Computational costs of RTMC subroutine.

ルーチン名	計算コスト (%)	処理内容
LGEVAL	49.7	二値理論式の評価を行う
CHKCEL	34.7	点がセルに含まれる点であるか否かをチェック
TRACE	5.2	視点へ向かう光線が他の物体に遮蔽されるかをチェック
DLOWP	2.5	４次方程式の解を求める
SURFGT	2.3	光線とトーラスの交点を計算する
SURFP	1.8	光線と平面の交点を計算する
CROSS2	1.7	光線が交差するセル、交点の座標、法線ベクトルを計算する
SURFGQ	1.1	光線と一般曲面との交点を計算する
RGBOUT	0.5	画像データを出力する
CHKBVL	0.3	光線とバウンダリ・ボリュームの交差テストを行う
RAY	0.1	光線追跡コントロール・ルーチン

（注）前出の表中の７番のケースのコスト

Table 5.3 Boundary volume data.

1	-7975.	-1550.	0.	4450.
2	-7950.	+1550.	0.	4450.
3	-8250.	-7922.	0.	1550.
4	-5650.	-7922.	0.	1550.
5	-10180.	-4665.	0.	1360.
6	-10180.	-3715.	0.	1360.
7	-13325.	-5050.	1178.	2300.
8	-13067.	-3850.	1178.	2300.
9	-13675.	-4975.	0.	2330.
10	-10700.	-10150.	0.	3200.
11	-11534.	-8928.	0.	1040.
12	-9715.	-10938.	0.	735.
13	-8099.	-11850.	0.	4420.
14	-10015.	-6520.	0.	800.
15	-6888.	0.	0.	1160.
16	-5385.	-11380.	0.	1750.
17	-12000.	3000.	1382.	250.

6. おわりに

本研究においては、光線追跡法による画像生成と粒子輸送モンテカルロ・コードの類似性に着目し、モンテカルロ法を適用した光線追跡画像生成プログラムを開発した。

このプログラムは、(1) 光線を光源から順方向に追跡することによって、光の伝播に忠実な画像を生成する、(2) モンテカルロ・コードに導入されている分散低減法を適用することにより計算の高速化を図った、さらに、(3) 高速モンテカルロ装置を用いたベクトル並列処理による計算の高速化を行った、といった特徴を有する。

原研の研究用原子炉 JRR-3M 一次区画内の主要機器の可視化により、開発したプログラムの性能評価を行い、従来の光線追跡法に比較して、より写実性の高い画像が生成できることを確認した。すなわち、本研究により、従来の光線追跡画像生成法に対する順方向光線追跡画像生成法の優位性が明らかになった。また、計算の高速化に関しても、分散低減法、バウンダリ・ボリュームの導入により計算時間を約 50 分の 1 に削減、さらに、高速モンテカルロ装置上でのベクトル並列処理によりスカラ処理に対しておよそ 19.5 倍の速度向上率（領域判定パイプラインの効果を含む）を達成した。これらの知見は、画像生成における高速化の観点からも有用である。

一方、高速化において十分な成果を得たものの、依然 (これでもなお)、写実性の高い画像を得るためには、従来手法と比較し、100 ～ 1,000 倍の処理時間を必要とする。この膨大な処理時間のさらなる高速化が今後の課題である。

謝　　辞

本研究実施にあたり、計算科学技術推進センター浅井センター長には、数々の貴重なる助言を頂いた。ここに深謝いたします。また、同センター相川次長には、本稿査読の際、細部にわたり貴重なる助言を頂いた。同じく、ここに深謝いたします。

6. おわりに

本研究においては、光線追跡法による画像生成と粒子輸送モンテカルロ・コードの類似性に着目し、モンテカルロ法を適用した光線追跡画像生成プログラムを開発した。

このプログラムは、(1) 光線を光源から順方向に追跡することによって、光の伝播に忠実な画像を生成する、(2) モンテカルロ・コードに導入されている分散低減法を適用することにより計算の高速化を図った、さらに、(3) 高速モンテカルロ装置を用いたベクトル並列処理による計算の高速化を行った、といった特徴を有する。

原研の研究用原子炉 JRR-3M 一次区画内の主要機器の可視化により、開発したプログラムの性能評価を行い、従来の光線追跡法に比較して、より写実性の高い画像が生成できることを確認した。すなわち、本研究により、従来の光線追跡画像生成法に対する順方向光線追跡画像生成法の優位性が明らかになった。また、計算の高速化に関しても、分散低減法、バウンダリ・ボリュームの導入により計算時間を約 50 分の 1 に削減、さらに、高速モンテカルロ装置上でのベクトル並列処理によりスカラ処理に対しておよそ 19.5 倍の速度向上率（領域判定パイプラインの効果を含む）を達成した。これらの知見は、画像生成における高速化の観点からも有用である。

一方、高速化において十分な成果を得たものの、依然 (これでもなお)、写実性の高い画像を得るためには、従来手法と比較し、100 ～ 1,000 倍の処理時間を必要とする。この膨大な処理時間のさらなる高速化が今後の課題である。

謝　　辞

本研究実施にあたり、計算科学技術推進センター浅井センター長には、数々の貴重なる助言を頂いた。ここに深謝いたします。また、同センター相川次長には、本稿査読の際、細部にわたり貴重なる助言を頂いた。同じく、ここに深謝いたします。

参考文献

[1] Catmull, E.: Computer Display of Curved Surfaces, *Proc. IEEE Conf. Comput. Graphics Pattern Recognition Data Struct.*, pp.11(1975).

[2] 矢島, 栗原：Z-バッファ・アルゴリズム, 日経ＣＧ, 1987 年 12 月号, pp.122-128(1987).

[3] Cohen, M. F. et al. : An Efficient Radiosity Approach for Realistic Image Synthesis, *IEEE Computer Graphics and Applications*, vol.6, no.3, pp.26-35(1986).

[4] Whitted, T. : An Improved Illumination Model for Shaded Display, *Comm. ACM*, vol.23, no.6, pp.343-349(1980).

[5] 藤本：自然物体をリアルに表現する双方向レイ・トレーシング法, 日経ＣＧ, 1987 年 6 月号, pp.110-120(1987).

[6] Pattanaik, S. N. and Mudur, S. P. : Adjoint Equations and Random Walks for Illumination Computation, *ACM Trans. Graphics*, vol.14, no.1, pp.77-102(1995).

[7] Cook R. L. et al. : Distributed Ray Tracing, *Computer Graphics*, vol.18, no.3, pp.137-146(1984).

[8] Marks W. B. : Difference Spectra of the Visual Pigments in Single Goldfish Cones, Baltimore, Johns Hopkins Univ.(1963).

[9] Tomita T. et al. : Spectral Response Curves of Single Cones in the Carps, *Vision Res.*, vol.7, pp.519-531(1967).

[10] 千葉, 村岡：レイトレーシングＣＧ入門, サイエンス社 (1990).

[11] Phong, B. T. : Illumination for Computer Generated Pictures, *Comm. ACM*, vol.18, no.8, pp.311-317(9175).

[12] 浅井ほか：原子力知能化システム技術の研究（人間動作シミュレーション・プログラム）平成 3 年度作業報告, *JAERI-M 90-060*, 日本原子力研究所 (1990).

[13] Weghorst, H. et al. : Improved Computational Methods for Ray Tracing, *ACM Trans. Graphics*, vol.3, no.1, pp.52-69 (1984).

[14] Higuchi, K. et al. : Development of Monte Carlo Machine for Particle Transport Problem, *J. Nucl. Sci. Technol.*, Vol. 32, No. 10(1995).

付録1 高速モンテカルロ装置の概要

高速モンテカルロ装置は、粒子輸送モンテカルロ・コード（以下、MCコード）高速化のために開発された専用計算機である。既存のスーパーコンピュータをベース・プロセッサとして、新規開発されたハードウェアが付加され、また、既存のハードウェアの一部が改良されている。

以下に粒子輸送モンテカルロ・コード高速化のために新規開発あるいは改良された高速モンテカルロ装置のハードウェア上の特長について述べる。

(1) モンテカルロ・パイプライン

本装置は、粒子輸送モンテカルロ計算において多用され、かつ、従来のベクトル計算機においてベクトル処理が困難な処理を高速化するため、モンテカルロ・パイプラインと呼ばれる3種類の特殊パイプラインを搭載している。各モンテカルロ・パイプラインの概念を以下に示す。

(a) 幾何形状パイプライン

MCコードにおいては、2次あるいは4次曲面の論理演算によって表現された3次元領域中の粒子を追跡する。この追跡計算において、粒子の現在位置から次の境界までの距離を計算する際、その領域を構成する曲面の種類によって粒子を分類する必要があるが、この処理は、ベクトル化不可能な多分岐の条件分岐文によって行われる。本パイプラインは、従来スカラ処理によって行われていたこの種のDOループを高速処理する。

(b) 事象分類パイプライン

MCコードのベクトル処理においては、散乱や吸収といった事象ごとに粒子を集め、ベクトル処理を行う。本パイプラインは、粒子を事象によって高速に分類し、粒子バンクを作成する。

(c) 領域判定パイプライン

2次あるいは4次曲面の論理演算によって表現された3次元領域において、粒子を追跡する際、ある点がどの領域に含まれるかを判定する処理が必要である。本パイプラインは、この処理を行うための二値論理式の評価を高速に行う。

(2) 強化されたロード／ストア・パイプライン

MCコードのベクトル処理においては、複数の粒子を同時に追跡する。計算において、各粒子の属性データは間接番地によって参照される。ところが、間接番地参照は、連続番地参照と異なり、主記憶からベクトル・レジスタへのデータ転送が遅く、このため、演算パイプラインに遊びが生じ、ベクトル・プロセッサの性能が低下する。本装置においては、間接番地参照データの転送を行うロード／ストア・パイプラインの能力が改造前のベース・プロセッサと比較し、8倍に強化されている。

(3) 並列処理機能

粒子輸送問題においては、各粒子に関する処理は独立であり、この独立性を利用した並列処理による高速化が可能である。本装置においては、４台のプロセッサを用いた並列処理が可能である。並列処理には、サブルーチン単位で並列処理を行うマクロ・タスク機能とＤＯループを並列処理するマイクロ・タスク機能の２種類がある。

付録2 ＲＴＭＣコードの説明

2.1 ＲＴＭＣコードの構成

本作業で作成したＲＴＭＣコードは、入力データの読み込み等を行う前処理部 (priprc)、光源よりサンプリングされた光線の追跡を行う光線追跡部 (ray)、光線追跡の結果から画像を出力する後処理部 (preprc) の３つで構成されている。ＲＴＭＣコードの構成を図 2.1 に示す。

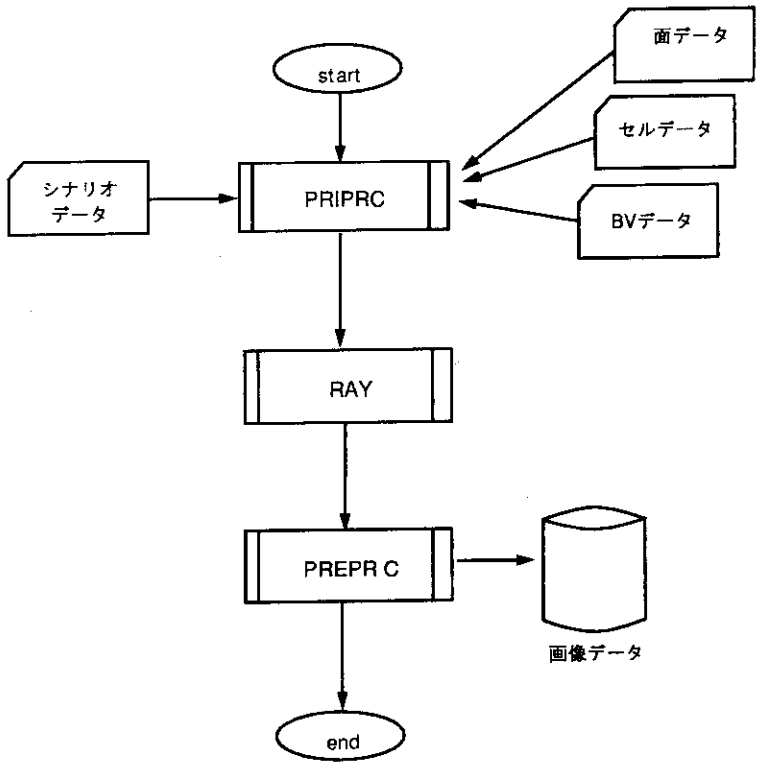

Fig. 2.1 Calculation Flow of the RTMC Code.

2.2 前処理部の説明

前処理部は、（1）入力データの読み込み、（2）スクリーンの計算 、（3）画素に関する配列の初期化、を行う。図 2.2 に前処理部の木構造を示す。各ルーチンは次のような処理を行う。

input 入力データの処理に関するコントロール・ルーチン

readc セルデータの読み込みを行う。

reads サーフェイスデータの読み込みを行う。

readb バウンダリ・ボリュームデータの読み込みを行う。

readd シナリオデータの読み込みを行う。

mkcel セル数をカウントし、セル番号、ポインタを配列に格納する。

mkbvl バウンダリ・ボリューム数をカウントし、バウンダリ・ボリューム番号等を配列に格納する。

mkdata シナリオデータの光源カード等の枚数をカウントする。

chekcs セルデータに含まれる補集合データを展開する。

items 各カードの空白で区切られる１つのデータを読み込む。

nextit items で読み込んだデータを、適当な配列に格納する。

namchg セル、面番号が定義されているかを確認する。

nxtsym nxtsym(hs,hd,ir,it,iu,ic)
hs の中からhs で区切られるデータの開始、終了ポインタit 及びiu を返す。

kdata データの型を返す。
0:character, 1:real or double precision, 2:integer

iblock シナリオデータ、面の方程式のニーモニックを定義する。

mkscrn スクリーンを定義する。

screen スクリーンの縦及び横方向のベクトルを計算する。

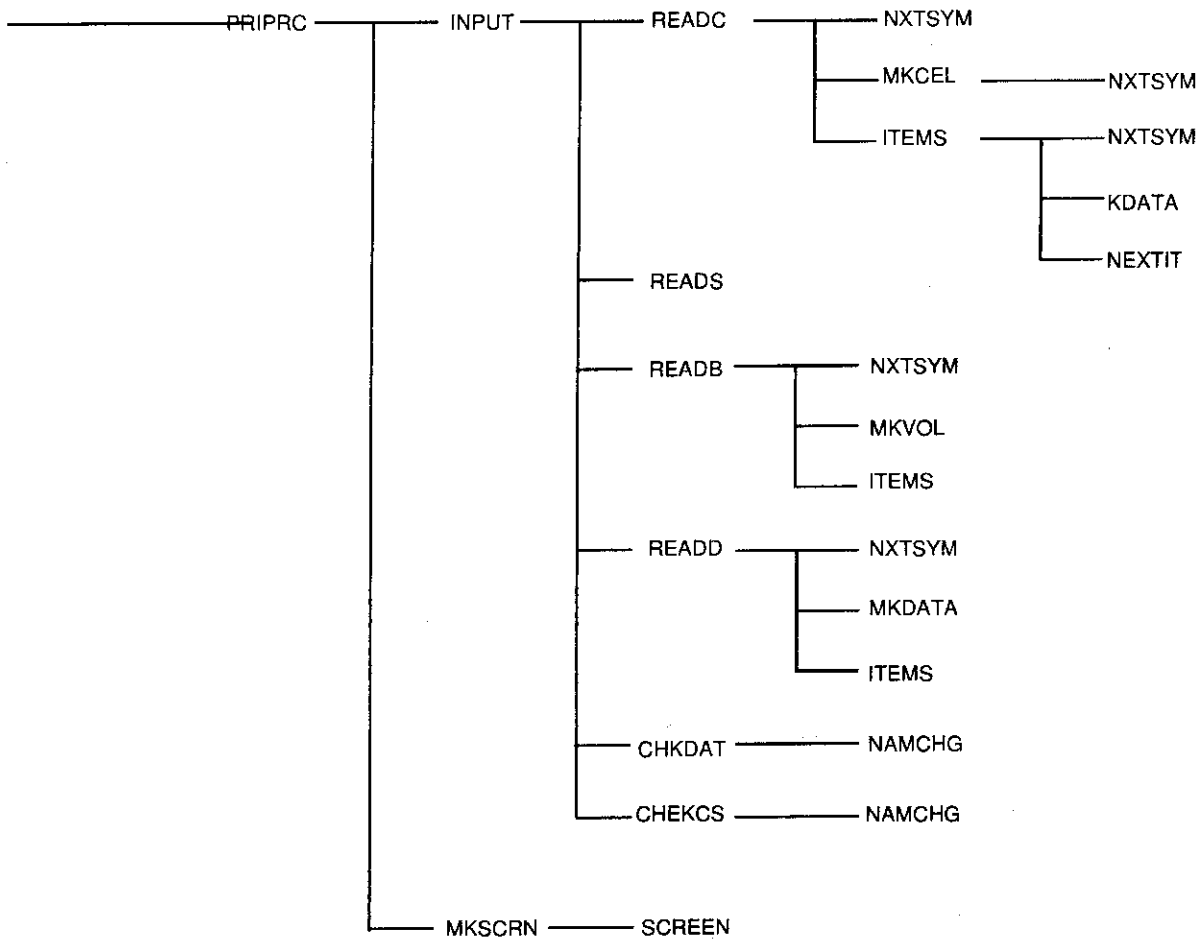

Fig. 2.2 前処理部の木構造

2.3 光線追跡部の説明

光線追跡部の処理は、(1) 光源から光線をサンプリングする、(2) 光線と形状の交差判定、(3) 衝突のシミュレーション、(4) 衝突点から視点 (画素) への寄与を計算する、である。図 2.3 に光源追跡部の木構造を示す。各ルーチンは次のような処理を行う。

ray 光線追跡部のコントロール・ルーチン

source 光線発生コントロール・ルーチン

sourc1 平行光線 を発生させる。

sourc2 点光源から光線を発生させる。

sourc3 線光源から光線を発生させる。

scross 光線と全領域を含むバウンダリ・ボリュームとの交点を求める。

cross2 光線と領域との交点を求める。

chkcel 交点が真にセルに含まれているかをチェックする。

lgeval 二値論理演算

angle 光線の鏡面反射光の方向ベクトルを求める。

trace 視点に向かう光線が他の物体で遮蔽されていかどうかをチェックする。

colset 各画素への寄与を求め、保存する。

surfxx 直線と平面との交点を求める。

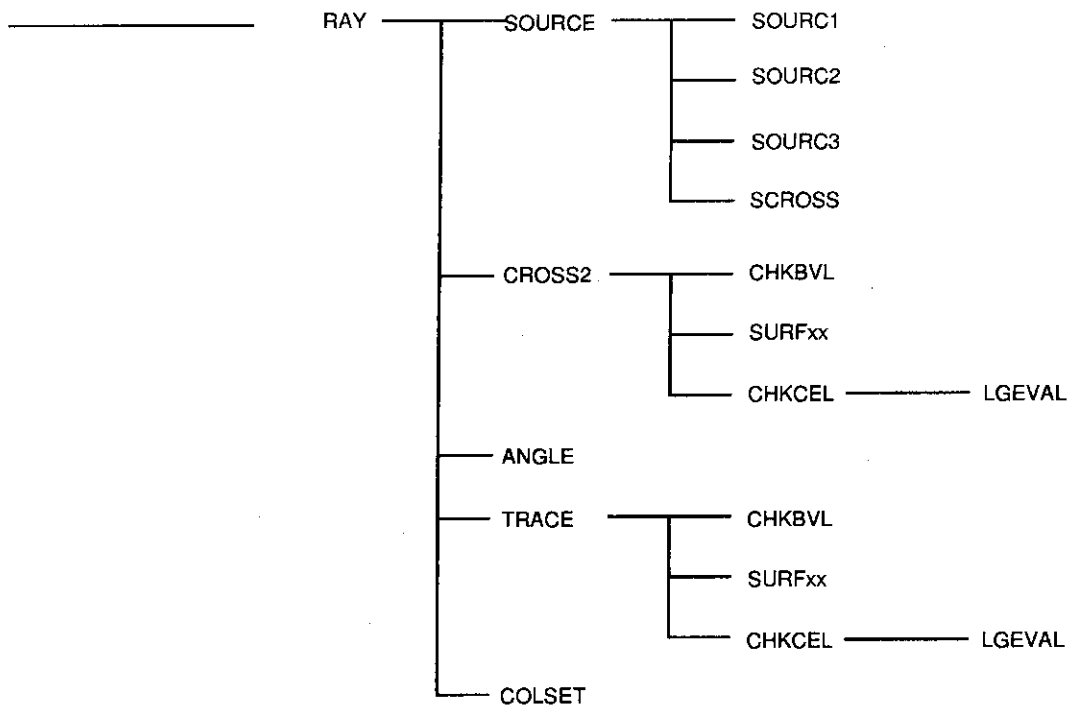

Fig. 2.3 光源追跡部の木構造

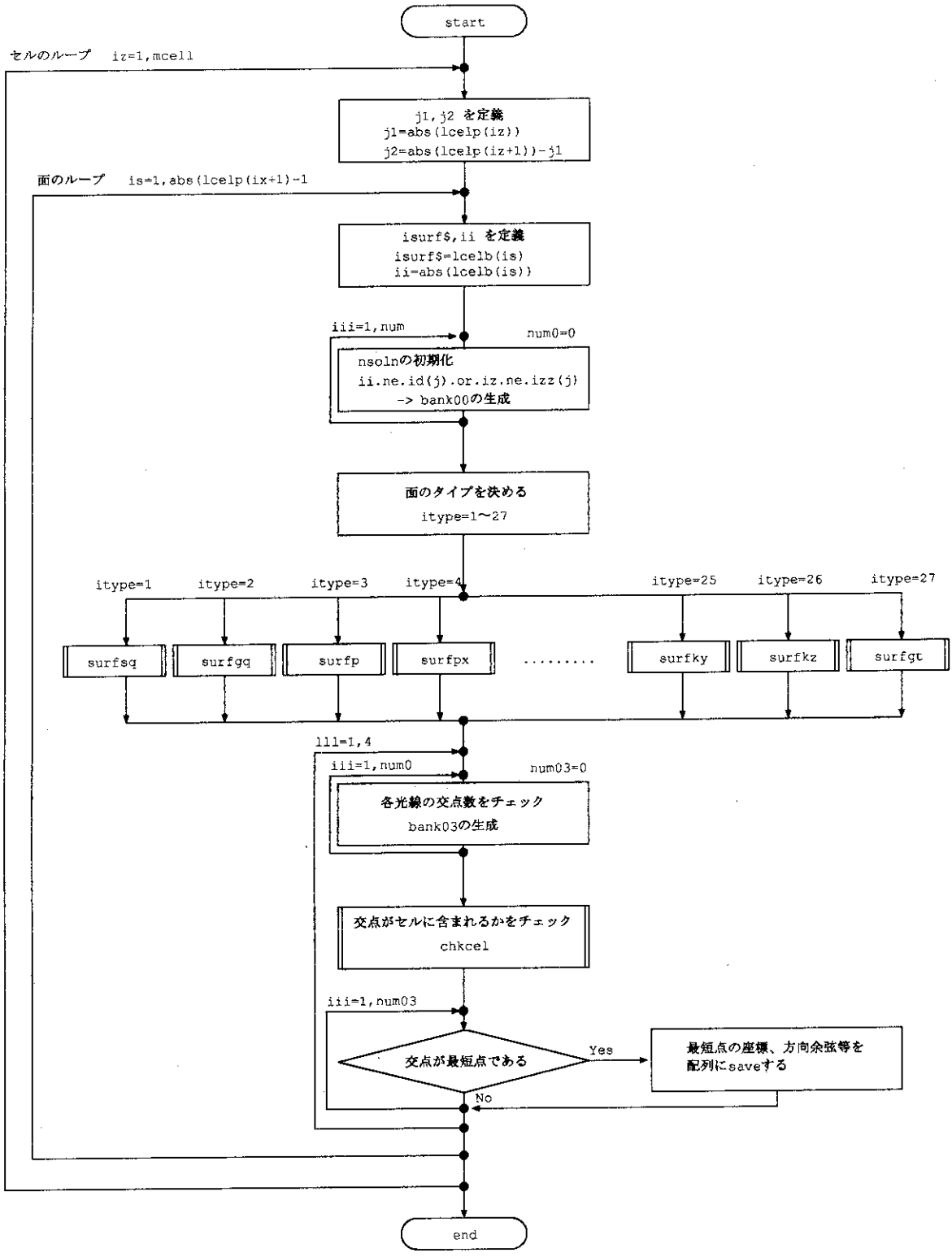

Fig. 2.4 cross のフロー

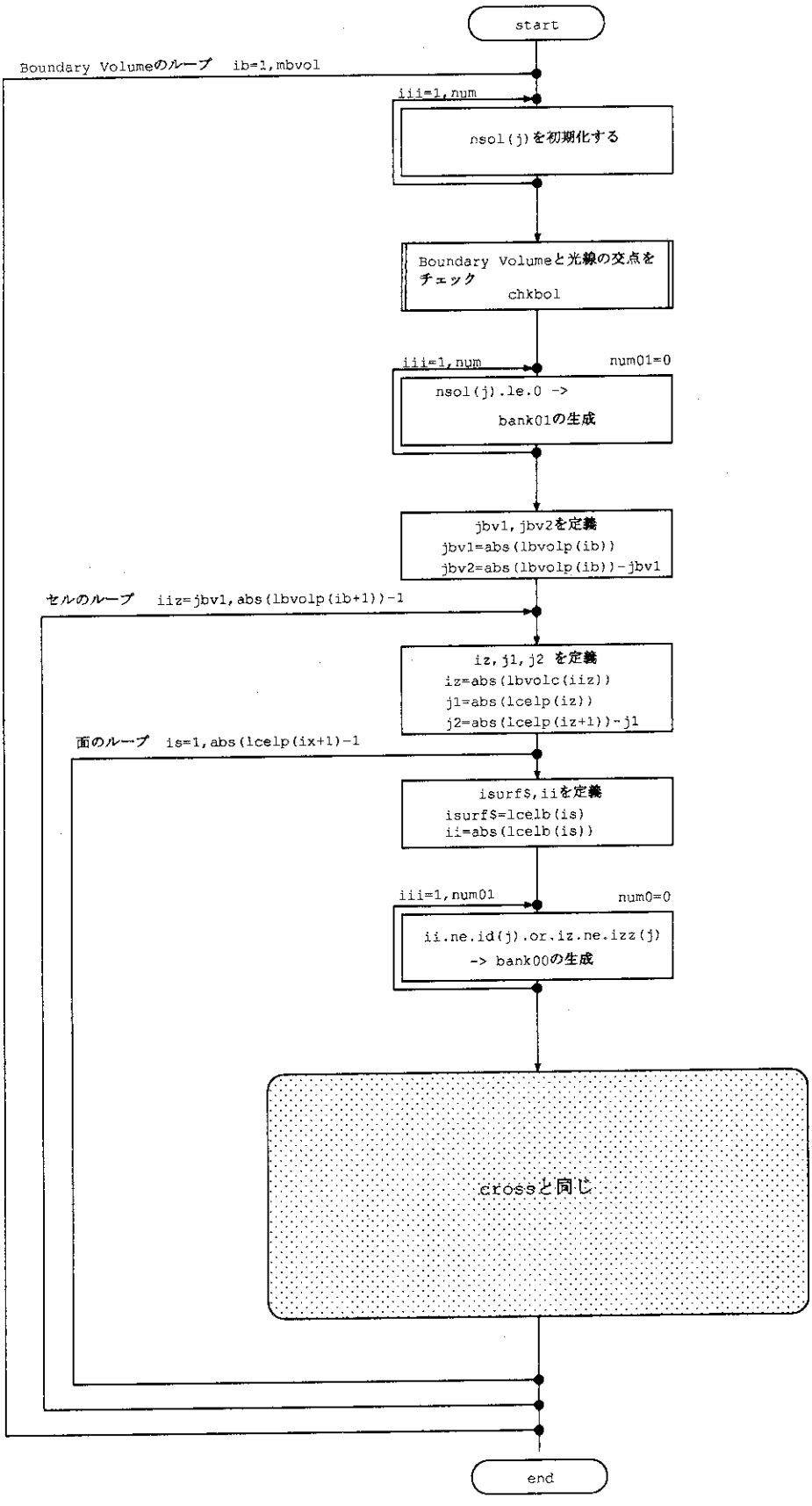

Fig. 2.5 cross2 のフロー

2.4 後処理部の説明

後処理部は、寄与エネルギーをＲＧＢ値に変換し、画像データを出力する。図 2.6 に後処理部の木構造を示す。各ルーチンは次のような処理を行う。

preprc 後処理部のコントロール・ルーチン

rgbset 寄与エネルギーをＲＧＢ値に変換

rgbout 画像を出力する

Fig. 2.6 後処理部の木構造

2.4.1 ＲＧＢ値への変換

光線追跡部で求めた各画素における寄与エネルギーを次のように離散化してＲＧＢ値へ変換する。

$$0 \leq Energy < E_1 \rightarrow RGB = 0$$

$$E_1 \leq Energy < E_2 \rightarrow RGB = 1$$

$$E_2 \leq Energy < E_3 \rightarrow RGB = 2$$

$$\vdots$$

$$Energy \geq E_{255} \rightarrow RGB = 255$$

ただし、

$$E_1 - 0 = E_2 - E_1 = E_3 - E_2 = \cdots\cdots = E_{i-1} - E_i = \cdots\cdots$$

すなわち、

$$cdiv = \frac{1}{E_i - E_{i-1}}$$

として、

$$RGB = int(\sqrt{Energy} * cdiv)$$

として、変換する。なお、$cdiv$ はパラメータで設定する。

2.5 プログラム・モジュールの説明

モジュール記述シート

プログラム名	RTMC
モジュール名	angle
モジュールタイプ	subroutine

機能

衝突点における入射光と法線ベクトルとの余弦値及び鏡面反射光の方向ベクトルを計算する。

呼び出し形式	angle(x,y,z,xn,yn,zn,u,v,w,costh,xr,yr,zr,num,bank)
呼出しモジュール	
被呼出しモジュール	ray

引数	型	配列	I/O	備考
x	R*8	(npart)	I	衝突点の X 座標
y	R*8	(npart)	I	衝突点の Y 座標
z	R*8	(npart)	I	衝突点の Z 座標
xn	R*8	(npart)	I	衝突点の法線ベクトルの U 成分
yn	R*8	(npart)	I	衝突点の法線ベクトルの V 成分
zn	R*8	(npart)	I	衝突点の法線ベクトルの W 成分
u	R*8	(npart)	I	入射光の方向ベクトルの U 成分
v	R*8	(npart)	I	入射光の方向ベクトルの V 成分
w	R*8	(npart)	I	入射光の方向ベクトルの W 成分
costh	R*8	(npart)	I/O	方向ベクトルと法線ベクトル余弦値
xr	R*8	(npart)	I/O	鏡面反射光の方向ベクトルの U 成分
yr	R*8	(npart)	I/O	鏡面反射光の方向ベクトルの V 成分
zr	R*8	(npart)	I/O	鏡面反射光の方向ベクトルの W 成分
num	I*4	(npart)	I	光線数
bank	I*4	(npart)	I	バンク

インクルード

PARAM

備考

モジュール記述シート

プログラム名	RTMC
モジュール名	chkbvl
モジュールタイプ	subroutne

機能

光線とバウンダリ・ボリューム（ＢＶ）との交点を求める。

呼び出し形式	call chkbvl(x1,y1,z1,u,v,w,nsol,kkk,num,bank)
呼出しモジュール	
被呼出しモジュール	cross2,trace

引数	型	配列	I/O	備考
x1	R*8	(npart)	I	光線のＸ座標
y1	R*8	(npart)	I	光線のＹ座標
z1	R*8	(npart)	I	光線のＺ座標
u	R*8	(npart)	I	光線の方向ベクトルのＵ成分
v	R*8	(npart)	I	光線の方向ベクトルのＶ成分
w	R*8	(npart)	I	光線の方向ベクトルのＷ成分
nsol	I*4	(npart)	I/O	交点の個数
kkk	I*4		I	ＢＶの番号
num	I*4		I	光線数
bank	I*4	(npart)	I	バンク

インクルード

PARAM,BCARD

備考

バウンダリ・ボリュームは面数が最小の球体を採用

モジュール記述シート

プログラム名	RTMC
モジュール名	chkcel
モジュールタイプ	subroutine

機能

点が真にセルに含まれる点であるかどうかをチェックする。

呼び出し形式	chkcel(solx,soly,solz,nflag,iz,num,bank)
呼出しモジュール	lgeval
被呼出しモジュール	cross2,trace

引数	型	配列	I/O	備考
solx	R*8	(npart)	I	X 座標
soly	R*8	(npart)	I	Y 座標
solz	R*8	(npart)	I	Z 座標
nflag	I*4	(npart)	I/O	セル内の点であるかどうかのフラグ
iz	I*4		I	セル番号 (インデックス番号)
num	I*4		I	光線数
bank	I*4	(npart)	I	バンク

インクルード

PARAM,CELL,SNAME,GEOM,GEOMGT

備考

NFLAG(i)=0: セル内にある

NFLAG(i)=1: セル内にある

配列 LG(INSURF*NPART) には 0 または 1 が入る。配列 LLG をＬＧＥＶＡＬに渡す。

モジュール記述シート

プログラム名	**RTMC**
モジュール名	**chekcs**
モジュールタイプ	**subroutine**

機能

セルデータ内の冗長な括弧を取り除き、補集合#を展開する。

呼び出し形式	**chekcs**
呼出しモジュール	**namchg**
被呼出しモジュール	**input**

引数	型	配列	I/O	備考
なし				

インクルード

PARAM,CELL,SURF

備考

モジュール記述シート

プログラム名	RTMC
モジュール名	colset
モジュールタイプ	subroutine
機能	光線と物体の衝突点から視点への寄与を計算する。
呼び出し形式	colset(duu,dvv,dww,intent,costh,wt,eng1,eng2,eng3, colr,colg,colb,colr2,colg2,colb2,ss, ncolor,izsave,num,bank)
呼出しモジュール	
被呼出しモジュール	ray

引数	型	配列	I/O	備考
duu	R*8	(npart)	I	視点への方向ベクトルの U 成分
dvv	R*8	(npart)	I	視点への方向ベクトルの V 成分
dww	R*8	(npart)	I	視点への方向ベクトルの W 成分
intent	I*4	(npart)	I	
costh	R*8	(npart)	I	光線と法線ベクトルの余弦値
wt	R*8	(npart)	I	光線のウエイト
eng1	R*8	(npart)	I	光線のエネルギー（赤）
eng2	R*8	(npart)	I	光線のエネルギー（緑）
eng3	R*8	(npart)	I	光線のエネルギー（青）
colr	R*8	(mln,iyysiz)	I/O	寄与エネルギー（赤）の総和
colg	R*8	(mln,iyysiz)	I/O	寄与エネルギー（緑）の総和
colb	R*8	(mln,iyysiz)	I/O	寄与エネルギー（青）の総和
colr2	R*8	(mln,iyysiz)	I/O	寄与エネルギー（赤）の二乗の総和
colg2	R*8	(mln,iyysiz)	I/O	寄与エネルギー（緑）の二乗の総和
colb2	R*8	(mln,iyysiz)	I/O	寄与エネルギー（青）の二乗の総和
ss	R*8	(npart)	I	画素に寄与した光線のウエイトの総和
ncolor	I*4	(mln,iyysiz)	I/O	画素に寄与した光線の総数
izsave	I*4	(npart)	I	セル番号
num	I*4	(npart)	I	光線数
bank	I*4	(npart)	I	バンク

インクルード

PARAM,CELL,DCARD,GRSIZE,SCSURF,PARTC2

備考　エネルギーに拡散反射率、鏡面反射率、透過率をかけて寄与エネルギーを計算。
寄与エネルギーは距離で減衰させる。寄与する画素を決定する。

モジュール記述シート

プログラム名	RTMC
モジュール名	cross
モジュールタイプ	subroutine

機能

光線が交差するセル、交点の座標、交点における法線ベクトルを計算する。

呼び出し形式	cross(num,bank)
呼出しモジュール	surfsq,surfgq,surfp,surfpx,surfpy,surfpz,surftz,surfty,surftx, surfso,surfs,surfsx,surfsy,surfsz,sufcpx,sufcpy,sufcpz,surfcx, surfcy,surfcz,sufkpx,sufkpy,sufkpz,surfkx,surfky,surfkz, surfgt,chkcel
被呼出しモジュール	ray

引数	型	配列	I/O	備考
num	I*4		I	光線数
bank	I*4	(npart)	I	バンク

インクルード

PARAM,CELL,SNAME,GEOM,PARTCL,PARTC2,SAVE

備考

バウンダリ・ボリューム適用以前のソース

j1:セル IZ のポインタ

J2:セル IZ のデータ数

光線の始点のあるセル番号≠チェックするセル番号ｏｒ

光線の始点のある面番号≠チェックする面番号

ならば bank00(num0)

交点あり ならば bank03(num03)

モジュール記述シート

プログラム名	**RTMC**
モジュール名	**cross2**
モジュールタイプ	**subroutine**

機能

光線が交差するセル、交点の座標、交点における法線ベクトルを計算する。

呼び出し形式	**cross(num,bank)**
呼出しモジュール	**surfsq,surfgq,surfp,surfpx,surfpy,surfpz,surftz,surfty,surftx, surfso,surfs,surfsx,surfsy,surfsz,sufcpx,sufcpy,sufcpz,surfcx, surfcy,surfcz,sufkpx,sufkpy,sufkpz,surfkx,surfky,surfkz, surfgt,chkcel,chkbvl**
被呼出しモジュール	**ray**

引数	型	配列	I/O	備考
num	**I*4**		**I**	光線数
bank	**I*4**	**(npart)**	**I**	バンク

インクルード

PARAM,CELL,SNAME,GEOM,PARTCL,PARTC2,SAVE, DCARD,BCARD

備考

j1:セル **IZ** のポインタ

J2:セル **IZ** のデータ数

光線の始点のあるセル番号≠チェックするセル番号ｏｒ

光線の始点のある面番号≠チェックする面番号

ならば **bank00(num0)**

交点あり ならば **bank03(num03)**

モジュール記述シート

プログラム名	**RTMC**
モジュール名	**iblock**
モジュールタイプ	**block data**

機能

シナリオデータ、サーフェイスデータのニーモニックを定義する。

呼び出し形式	**block data ibldat**
呼出しモジュール	
被呼出しモジュール	

インクルード

PARAM,DCARD,SNAME

備考

```
cname(1:11)
data cname( 1)/' '/
data cname( 2)/' '/
data cname( 3)/' '/
data cname( 4)/'view '/
data cname( 5)/'scrn '/
data cname( 6)/'scal '/
data cname( 7)/'so '/
data cname( 8)/'m '/
data cname( 9)/'imp '/
data cname(10)/'avol '/
data cname(11)/'hist '/
markv(1:27)
data markv( 1)/' SQ'/
data markv( 2)/' GQ'/
data markv( 3)/' P '/
data markv( 4)/' PX'/
data markv(24)/' KX'/
      .
      .
      .
```

モジュール記述シート

プログラム名	**RTMC**
モジュール名	**input**
モジュールタイプ	**subroutine**

機能

入力データ（サーフェイス、セル、バウンダリ・ボリューム、シナリオ）
を読み込む。

呼び出し形式	**input**
呼出しモジュール	**reads,readc,readd,readb,chekcs**
被呼出しモジュール	**priprc**

引数	型	配列	I/O	備考

インクルード

PARAM,CELL,DCARD,ITEMC,GRSIZE

備考

セルデータの最終ポインタを決める。

lcelp(mcell+1) = ncell + 1

モジュール記述シート

プログラム名	**RTMC**
モジュール名	**items**
モジュールタイプ	**subroutine**

機能

カードの空白で区切られた１つのデータを読み込む。

呼び出し形式	**items**
呼出しモジュール	**nxtsym,nextit,kdata**
被呼出しモジュール	**readc,readd,readb**

引数	型	配列	I/O	備考

インクルード

PARAM,ITEMC,CARD80

備考

MCNP4a を参考に作成。

モジュール記述シート

プログラム名	**RTMC**
モジュール名	**kdata**
モジュールタイプ	**function**

機能

hd のデータの型を返す。

呼び出し形式	**kdata(hd)**
呼出しモジュール	
被呼出しモジュール	**items,mkdata**

引数	型	配列	I/O	備考
hd	**A***	**(*)**	**I**	チェックするキャラクター配列

コモン

備考

0:character, 1:real or double precision , 2:integer

モジュール記述シート

プログラム名	**RTMC**
モジュール名	**lgeval**
モジュールタイプ	**function**

機能

二値論理式の評価を行う。

呼び出し形式	**lgeval(lg,n)**
呼出しモジュール	
被呼出しモジュール	**chkcel**

引数	型	配列	I/O	備考
lg	**I*4**	**(*)**	**I**	**0 または 1**
n	**I*4**		**I**	**lg の大きさ**

インクルード

備考

領域判定パイプラインと置換可能。

(: **1000001**

) : **1000002**

union : **1000003**

true : **1**

false : **0**

intersection : **implicit**

モジュール記述シート

プログラム名	TRMC
モジュール名	namchg
モジュールタイプ	function

機能

mm が定義されているセル、又は面であるかをチェックする。

呼び出し形式	namchg(mm,ji)
呼出しモジュール	
被呼出しモジュール	chekcs

引数	型	配列	I/O	備考
mm	I*4		I	処理タイプ、 1 ならばセル、2 ならば面
ji	I*4		I	チェックするセル番号または面番号

インクルード

PARAM,CELL,SAVE,CARD80

備考

MCNP4a を参考に作成。

モジュール記述シート

プログラム名	RTMC
モジュール名	nextit
モジュールタイプ	subroutine

機能

サブルーチン items で読み込んだデータを適当な配列に格納する。

呼び出し形式	nextit
呼出しモジュール	
被呼出しモジュール	readc,readd,readb

引数	型	配列	I/O	備考

インクルード

PARAM,CELL,DCARD,BCARD,
ITEMC,CARD80

備考

MCNP4a より

入力データが各カードで定義できる項目数を超えている場合はエラー出力をする。

モジュール記述シート

プログラム名	RTMC
モジュール名	nxtsym
モジュールタイプ	subroutine

機能

任意の文字で区切られるデータの開始及び終了ポインターを返す。

呼び出し形式	nxtsym(hs,hd,ir,it,iu,ic)
呼出しモジュール	
被呼出しモジュール	items,mkcel,mkdata,readc,readd,readb

引数	型	配列	I/O	備考
hs	A*	(*)	I	読み込まれているキャラクター配列
hd	A*	(*)	I	区切りのシンボル
ir	I*4		I	チェックを開始するポインタ
it	I*4		O	抜き出す文字列の開始ポインタ
iu	I*4		O	抜き出す文字列の終了ポインタ
ic	I*4		I	0 or 1 or 2

コモン

備考

例：call nxtsym(chi,' ',1,it,iu,1)

MCNP4a より

モジュール記述シート

プログラム名	**RTMC**
モジュール名	**mkdata**
モジュールタイプ	**subroutine**

機能

シナリオデータのタイプごとにデータ数のカウント等を行う。

呼び出し形式	**mkdata**
呼出しモジュール	**nxtsym,namchg**
被呼出しモジュール	**readd**

引数	型	配列	I/O	備考

コモン

PARAM,CELL,DCARD,ITEMS,CARD80

備考

$itye = 6$: 光源の数をカウント

$itye = 8$: セル属性データのカウント及びセル番号のセット

$itye = 10$: 全領域のバウンダリ・ボリュームのフラグのセット

モジュール記述シート

プログラム名	RTMC
モジュール名	mkscrn
モジュールタイプ	subroutine

機能

スクリーンを定義する。

スクリーンを表わす面の方程式の係数を計算する。

呼び出し形式	mkscrn
呼出しモジュール	screen
被呼出しモジュール	priprc

引数	型	配列	I/O	備考

インクルード

PARAM,CELL,DCARD,GRSIZE,SCSURF

備考

モジュール記述シート

プログラム名	**RTMC**
モジュール名	**main**
モジュールタイプ	**main routine**

機能

メインルーチン

呼び出し形式				
呼出しモジュール		**seedn,ranu21,priprc,ray,preprc**		
被呼出しモジュール				
引数	型	配列	I/O	備考

インクルード

PARAM,DCARD,COLOR

備考

(1) 各タスクの乱数の種を生成 (seedn)

(2) 乱数生成ベクトル処理で使用する係数を計算 (ranu21)

(3) 前処理 (priprc)

(4) 光線追跡 (ray)、並列処理される。

マクロタスク機能を使用 (ptfork,ptjoin)

(5) 後処理 (preprc)

モジュール記述シート

プログラム名	**RTMC**
モジュール名	**mkbvol**
モジュールタイプ	**subroutine**

機能

バウンダリ・ボリュームをカウントし、そのデータを配列に格納する。

呼び出し形式	**call mkbvol**
呼出しモジュール	**nxtsym**
被呼出しモジュール	**readb**

引数	型	配列	I/O	備考

インクルード

PARAM,CELL,BCARD,ITEMC,CARD80

備考

$itype = 3$

$mbvol = mbvol + 1$

モジュール記述シート

プログラム名	**RTMC**
モジュール名	**mkcel**
モジュールタイプ	**subroutine**

機能

セル数をカウントし、セル番号、ポインタを配列に格納する。

呼び出し形式	**mkcel**
呼出しモジュール	**nxtsym**
被呼出しモジュール	**readc**

引数	型	配列	I/O	備考

インクルード

PARAM,CELL,ITEMC,CARD80

備考

$mcell = mcell + 1$: セル総数のカウント

$ncelb(mcell) = icn$: セル番号を格納

$lcelp(mcell) = ncell + 1$: セルの先頭アドレスを格納

モジュール記述シート

プログラム名	**RTMC**
モジュール名	**preprc**
モジュールタイプ	**subroutine**

機能

後処理 (画像データの出力等) のコントロール・ルーチン

呼び出し形式	**preprc**
呼出しモジュール	**rgbset,rgbout**
被呼出しモジュール	**main**

引数	型	配列	I/O	備考

インクルード

PARAM,COLOR

備考

(1) 並列の場合、各タスクの計算結果を集計する。$maxcpu \geq 2$

(2) 各画素の寄与エネルギーの総和をＲＧＢ値に変換（ＲＧＢＳＥＴ）

(3) 画像を出力（ＲＧＢＯＵＴ）

モジュール記述シート

プログラム名	**RTMC**
モジュール名	**priprc**
モジュールタイプ	**subroutine**

機能

前処理 (入力データの読み込み等) のコントロール・ルーチン

呼び出し形式	**priprc**
呼出しモジュール	**input,mkscrn**
被呼出しモジュール	**main**

引数	型	配列	I/O	備考

インクルード

PARAM,ICOL,COLOR

備考

(1) 入力データの読み込み。

(2) スクリーンの定義。

(3) 配列 **(colr,colb,colb,colr2,colg2,colb2** など) の初期化。

モジュール記述シート

プログラム名	RTMC
モジュール名	ray
モジュールタイプ	subroutine

機能

光線追跡コントロールルーチン。

呼び出し形式	call ray(colr,colg,colb,colr2,colg2,colb2,ss, ncolor,ixr,rr,locr,nnpsp)
呼出しモジュール	source,cross,angle,trace,colset,ranu22
被呼出しモジュール	main

引数	型	配列	I/O	備考
colr	R*8	(mln,iyysiz)	I/O	寄与エネルギー（赤）の総和
colg	R*8	(mln,iyysiz)	I/O	寄与エネルギー（緑）の総和
colb	R*8	(mln,iyysiz)	I/O	寄与エネルギー（青）の総和
colr2	R*8	(mln,iyysiz)	I/O	寄与エネルギー（赤）の二乗の総和
colg2	R*8	(mln,iyysiz)	I/O	寄与エネルギー（緑）の二乗の総和
colb2	R*8	(mln,iyysiz)	I/O	寄与エネルギー（青）の二乗の総和
ss	R*8	(mln,iyysiz)	I/O	画素に寄与した光線のウエイトの総和
ncolor	I*4	(mln,iyysiz)	I/O	画素に寄与した光線の総数
ixr	I*4		I/O	乱数の種
rr	R*8	nrr	I/O	乱数
locr	I*4		I/O	乱数 rr の使用した数
nnpsp	I*4		I	追跡する光線数

インクルード

PARAM,CELL,DCARD,SCSURF,PARTCL,PARTC2,SAVE

備考

並列処理の場合、子ルーチンとなる。

モジュール記述シート

プログラム名	RTMC
モジュール名	readb
モジュールタイプ	subroutine

機能

バウンダリ・ボリュームデータを読み込む。

呼び出し形式	call readb
呼出しモジュール	nxtsym,mkbvol,items
被呼出しモジュール	input

引数	型	配列	I/O	備考
引数なし				

インクルード

PARAM,CELL,BCARD,CARD80

備考

モジュール記述シート

プログラム名	RTMC
モジュール名	readc
モジュールタイプ	subroutine

機能

セルデータの読み込みを行う。

呼び出し形式	readc
呼出しモジュール	nxtsym,mkcel,items
被呼出しモジュール	input

引数	型	配列	I/O	備考

インクルード

PARAM,CELL,ITEMC,CARD80

備考

カード（データの１行）がコメント、新カード、継続カード、ブランクカードのどれかを判断。

新カードの場合のみ、mkcelが呼び出される。

各カードの項目を読み込むためにitemsが呼び出される。

モジュール記述シート

プログラム名	RTMC
モジュール名	readd
モジュールタイプ	subroutine

機能

シナリオデータの読み込みを行う。

呼び出し形式	readd
呼出しモジュール	nxtsym,mkdata,items
被呼出しモジュール	input

引数	型	配列	I/O	備考

インクルード

PARAM,CELL,DCARD,CARD80

備考

モジュール記述シート

プログラム名	**RTMC**
モジュール名	**reads**
モジュールタイプ	**subroutine**

機能

サーフェイスデータの読み込みを行う。

呼び出し形式	**reads**
呼出しモジュール	
被呼出しモジュール	**input**

引数	型	配列	I/O	備考

インクルード

PARAM,SURF,GEOM,GEOMGT

備考

サーフェイスデータはＶＳＭＣコードと同じフォーマット

モジュール記述シート

プログラム名	RTMC
モジュール名	rgbout
モジュールタイプ	subroutine

機能

画像データを出力する。

呼び出し形式	rgbout(iscale)
呼出しモジュール	
被呼出しモジュール	preprc

引数	型	配列	I/O	備考
iscale	I*4		I	0 or 1

インクルード

PARAM,DCARD,GRSIZE,ICOL

備考

iscal が

0 : 光源強度が２５５を超える場合

1 : 光源強度が０から２５５までの場合

モジュール記述シート

プログラム名	RTMC
モジュール名	rgbset
モジュールタイプ	subrutine

機能

計算された各画素への寄与エネルギーをＲＧＢ値に変換する。

呼び出し形式	rgbset(colr,colg,colb,colr2,colg2,colb2, ss,ncolor)
呼出しモジュール	
被呼出しモジュール	preprc

引数	型	配列	I/O	備考
colr	R*8	(mln,iyysiz)	I/O	寄与エネルギー（赤）の総和
colg	R*8	(mln,iyysiz)	I/O	寄与エネルギー（緑）の総和
colb	R*8	(mln,iyysiz)	I/O	寄与エネルギー（青）の総和
colr2	R*8	(mln,iyysiz)	I/O	寄与エネルギー（赤）の二乗の総和
colg2	R*8	(mln,iyysiz)	I/O	寄与エネルギー（緑）の二乗の総和
colb2	R*8	(mln,iyysiz)	I/O	寄与エネルギー（青）の二乗の総和
ss	R*8	(mln.iyysiz)	I/O	画素に寄与した光線のウエイトの総和
ncolor	I*4	(mln,iyysiz)	I/O	画素に寄与した光線の総数

インクルード

PARAM,DCARD,GRSIZE,ICOL

備考

モジュール記述シート

プログラム名	RTMC
モジュール名	screen
モジュールタイプ	subroutine

機能

視線ベクトルからスクリーンの縦及び横方向のベクトルを計算する。

呼び出し形式	screen(xv,yv,zv,xf,yf,zf,uuu,vvv)
呼出しモジュール	
被呼出しモジュール	mkscrn

引数	型	配列	I/O	備考
xv	R*8		I	視点の X 座標
yv	R*8		I	視点の Y 座標
zv	R*8		I	視点の Z 座標
xf	R*8		I	スクリーンの X 座標
yf	R*8		I	スクリーンの Y 座標
zf	R*8		I	スクリーンの Z 座標
uuu	R*8	(3)	O	スクリーンの横方向のベクトル
vvv	R*8	(3)	O	スクリーンの縦方向のベクトル

コモン

備考

モジュール記述シート

プログラム名	RTMC
モジュール名	scross
モジュールタイプ	subroutine

機能

光線と全領域を包むバウンダリ・ボリュームとの交点を求める。

呼び出し形式	call scross(x1,y1,z1,u,v,w,nsol,num,bank)
呼出しモジュール	
被呼出しモジュール	source

引数	型	配列	I/O	備考
x1	R*8	(npart)	I	光線のＸ座標
y1	R*8	(npart)	I	光線のＹ座標
z1	R*8	(npart)	I	光線のＺ座標
u	R*8	(npart)	I	光線の方向ベクトルのＵ成分
v	R*8	(npart)	I	光線の方向ベクトルのＶ成分
w	R*8	(npart)	I	光線の方向ベクトルのＷ成分
nsol	R*8	(npart)	I/O	交点の数
num	I*4		I	光線数
bank	I*4	(npart)	I	バンク

インクルード

PARAM,DCARD

備考

バウンダリ・ボリュームは面数が最小の球体を採用

モジュール記述シート

プログラム名	RTMC
モジュール名	source
モジュールタイプ	subroutine

機能

光線を発生する。

呼び出し形式	source(ixr,rr,locr,i,num,bank)
呼出しモジュール	sourc1,sourc2,sourc3,scross
被呼出しモジュール	ray

引数	型	配列	I/O	備考
ixr	I*4		I/O	乱数の種
rr	R*8	(nrr)	I/O	乱数
locr	I*4		I/O	乱数ｒｒの使用した数
i	I*4		I	光源の番号
num	I*4		I	発生させる光線の数
bank	I*4	(npart)	I	バンク

インクルード

PARAM,DCARD,PARTCL,

備考

$light(i)$ が

1：平行光線

2：点光源

3：線光線

$nflg\_c8+ \neq 0$ は全領域を包み込むバウンダリ・ボリュームが
定義されている（シナリオデータで AVOL が指定されている）ことを表わす。
平行光線のときnflgs(j)=1

モジュール記述シート

プログラム名	RTMC
モジュール名	sourc1
モジュールタイプ	subroutine

機能

平行光線の場合に呼び出され、光線を発生する。

呼び出し形式	sourc1(ixr,rr,locr,x0,y0,z0,u,v,w,iii,num,bank)
呼出しモジュール	ranu22
被呼出しモジュール	source

引数	型	配列	I/O	備考
ixr	I*4		I/O	乱数の種
rr	R*8	(nrr)	I/O	乱数
locr	I*4		I/O	乱数ｒｒの使用した数
x0	R*8	(npart)	O	初期 X 座標
y0	R*8	(npart)	O	初期 Y 座標
z0	R*8	(npart)	O	初期 Z 座標
u	R*8	(npart)	O	方向ベクトルの U 成分
v	R*8	(npart)	O	方向ベクトルの V 成分
w	R*8	(npart)	O	方向ベクトルの W 成分
iii	I*4		I	光源の番号
num	I*4		I	光線数
bank	I*4	(npart)	I	バンク

インクルード

PARAM,DCARD

備考

光線を発生させる円盤の半径、変換する天頂角、方位角を入力データ

で与える

モジュール記述シート

プログラム名	RTMC
モジュール名	sourc2
モジュールタイプ	subroutine

機能

点光源の場合に呼び出されて、光線を発生する。

呼び出し形式	sourc2(ixr,rr,locr,x0,y0,z0,u,v,w,iii,num,bank)
呼出しモジュール	ranu22
被呼出しモジュール	source

引数	型	配列	I/O	備考
ixr	I*4		I/O	乱数の種
rr	R*8	(nrr)	I/O	乱数
locr	I*4		I/O	乱数 rr の使用した数
x0	R*8	(npart)	O	初期 X 座標
y0	R*8	(npart)	O	初期 Y 座標
z0	R*8	(npart)	O	初期 Z 座標
u	R*8	(npart)	O	方向ベクトルの U 成分
v	R*8	(npart)	O	方向ベクトルの V 成分
w	R*8	(npart)	O	方向ベクトルの W 成分
iii	I*4		I	光源の番号
num	I*4		I	光源数
bank	I*4	(npart)	I	バンク

インクルード

PARAM,DCARD

備考

モジュール記述シート

プログラム名	RTMC
モジュール名	sourc3
モジュールタイプ	subroutine

機能

線光源の場合に呼び出されて、光線を発生する。

呼び出し形式	sourc3(ixr,rr,locr,x0,y0,z0,u,v,w,iii,num,bank)
呼出しモジュール	ranu22
被呼出しモジュール	source

引数	型	配列	I/O	備考
ixr	I*4		I/O	乱数の種
rr	R*8	(nrr)	I/O	乱数
locr	I*4		I/O	乱数 rr の使用した数
x0	R*8	(npart)	O	初期 X 座標
y0	R*8	(npart)	O	初期 Y 座標
z0	R*8	(npart)	O	初期 Z 座標
u	R*8	(npart)	O	方向ベクトルの U 成分
v	R*8	(npart)	O	方向ベクトルの V 成分
w	R*8	(npart)	O	方向ベクトルの W 成分
iii	I*4		I	光源の番号
num	I*4		I	光源数
bank	I*4	(npart)	I	バンク

インクルード

備考

モジュール記述シート

プログラム名	**RTMC**
モジュール名	**surfgq**
モジュールタイプ	**subroutine**

機能

光線と円柱、円錐、楕円、双曲面、放物線との交点及び交点における面の法線ベクトルを計算する。

呼び出し形式	**surfgq(x1,y1,z1,uu,vv,ww,solx,soly,solz, solxn,solyn,solzn,nsol,nsurf,num,bank)**
呼出しモジュール	
被呼出しモジュール	**cross**

引数	型	配列	I/O	備考
x1	**R*8**	**(npart)**	**I**	直線の現在値の **X** 座標
y1	**R*8**	**(npart)**	**I**	直線の現在値の **Y** 座標
z1	**R*8**	**(npart)**	**I**	直線の現在値の **Z** 座標
uu	**R*8**	**(npart)**	**I**	直線の方向ベクトルの **U** 成分
vv	**R*8**	**(npart)**	**I**	直線の方向ベクトルの **V** 成分
ww	**R*8**	**(npart)**	**I**	直線の方向ベクトルの **W** 成分
solx	**R*8**	**(npart,4)**	**O**	交点の **X** 座標
soly	**R*8**	**(npart,4)**	**O**	交点の **Y** 座標
solz	**R*8**	**(npart,4)**	**O**	交点の **Z** 座標
solxn	**R*8**	**(npart,4)**	**O**	交点における面の法線ベクトルの **U** 成分
solyn	**R*8**	**(npart,4)**	**O**	交点における面の法線ベクトルの **V** 成分
solzn	**R*8**	**(npart,4)**	**O**	交点における面の法線ベクトルの **W** 成分
nsol	**I*4**	**(npart)**	**O**	交点数
nsurf	**I*4**		**I**	面番号
num	**I*4**		**I**	光線数
bank	**I*4**	**(npart)**	**I**	バンク

インクルード

PARAM,GEOM

備考

$$Ax^2 + By^2 + Cz^2 + Dxy + Eyz + Fzx + Gx + Hy + Iz + J = 0$$

モジュール記述シート

プログラム名	RTMC
モジュール名	trace
モジュールタイプ	subroutine

機能

視点へ向かう光線が他の物体によって遮蔽されるかどうかをチェックする。

呼び出し形式	trace(kd,kz,xx,yy,zz,uu,vv,ww,kflag,num,bank)
呼出しモジュール	surfsq,surfgq,surfp,surfpx,surfpy,surfpz,surftz,surfty,surftx, surfso,surfs,surfsx,surfsy,surfsz,sufcpx,sufcpy,sufcpz,surfcx, surfcy,surfcz,sufkpx,sufkpy,sufkpz,surfkx,surfky,surfkz, surfgt,chkcel,chkbvl
被呼出しモジュール	ray

引数	型	配列	I/O	備考
kd	I*4	(npart)	I	面番号
kz	I*4	(npart)	I	セル番号
xx	R*8	(npart)	I	衝突点の X 座標
yy	R*8	(npart)	I	衝突点の X 座標
zz	R*8	(npart)	I	衝突点の X 座標
uu	R*8	(npart)	I	視点への方向ベクトルの U 成分
vv	R*8	(npart)	I	視点への方向ベクトルの V 成分
ww	R*8	(npart)	I	視点への方向ベクトルの W 成分
kflag	I*4	(npart)	O	遮蔽物 の有無に関するフラグ
num	I*4		I	光線数
bank	I*4	(npart)	I	バンク

インクルード

PARAM,DCARD,CELL,SNAME,GEOM,BCARD

備考

kflag(i)=0:遮蔽物がある

kflag(i)=1:遮蔽物はない

モジュール記述シート

プログラム名	**RTMC**
モジュール名	乱数ルーチン
モジュールタイプ	**subroutine**

機能

乱数関連ルーチン

備考

seedn

ranu21

ranu22

asm01

モジュール記述シート

プログラム名	RTMC
モジュール名	BCARD
モジュールタイプ	common

機能

バウンダリ・ボリュームの情報を格納

被呼出しモジュール	chkbvl,cross2,mkbvol,nextit,readb,trace mkscrn,nextit,ray,readc,readd,trace,namchg

common/bcard/xbarb(maxbv),ybarb(maxbv),zbarb(maxbv),radb(maxbv)

common/bcard/mbvol

変数	型	配列	備考
xbarb	R*8	(maxbv)	各バウンダリ・ボリュームの中心のX座標
ybarb	R*8	(maxbv)	各バウンダリ・ボリュームの中心のY座標
zbarb	R*8	(maxbv)	各バウンダリ・ボリュームの中心のZ座標
radb	R*8	(maxbv)	各バウンダリ・ボリュームの半径

備考

モジュール記述シート

プログラム名	**RTMC**
モジュール名	**CARD80**
モジュールタイプ	**common**

機能

データの１行を読み込んでその情報を適当な変数に格納する。

被呼出しモジュール	**items,mkcel,mkdata,nextit,readc,readd,namchg**

character chi*80

common/card80/itype,nwc,chi

変数	型	配列	備考
itype	**I*4**		カードタイプ
nwc	**I*4**		カードデータの項目番号
chi	**A*80**		カードデータ

備考

モジュール記述シート

プログラム名	RTMC
モジュール名	CELL
モジュールタイプ	common

機能

セルデータの情報を格納

被呼出しモジュール	chekcs,chkcel,colset,cross,cross2,input,mkcel, mkbvol, mkdata, mkscrn,nextit,ray,readb, readc, readd,trace,namchg

common/cell/mcell,ncell,ncelb(maxc),lcelp(maxc),lcelb(maxc)

変数	型	配列	備考
mcell	I*4		セル数
ncell	I*4		セルデータ数
ncelb	I*4	(maxc)	セル番号
lcelp	I*4	(maxc)	各セルデータのポインタ
lcelb	I*4	(maxc)	セルの幾何形状、符号付き面番号とブーリアン演算子

備考

モジュール記述シート

プログラム名	RTMC
モジュール名	COLOR
モジュールタイプ	common

機能

各画素の情報を格納する。

被呼出しモジュール	colset,priprc,rgbout,rgbset

COMMON/COLOR/COLR(MLN,IYYSIZ,MAXCPU)
COMMON/COLOR/COLG(MLN,IYYSIZ,MAXCPU)
COMMON/COLOR/COLB(MLN,IYYSIZ,MAXCPU)
COMMON/COLOR/COLR2(MLN,IYYSIZ,MAXCPU)
COMMON/COLOR/COLG2(MLN,IYYSIZ,MAXCPU)
COMMON/COLOR/COLB2(MLN,IYYSIZ,MAXCPU)
COMMON/COLOR/SS(MLN,IYYSIZ,MAXCPU)
COMMON/COLOR/NCOLOR(MLN,IYYSIZ,MAXCPU)

変数	型	配列	備考
colr	R*8	(mln,iyysiz,maxcpu)	寄与エネルギー (赤) の総和
colg	R*8	(mln,iyysiz,maxcpu)	寄与エネルギー (緑) の総和
colb	R*8	(mln,iyysiz,maxcpu)	寄与エネルギー (青) の総和
colr2	R*8	(mln,iyysiz,maxcpu)	寄与エネルギー (赤) の二乗の総和
colg2	R*8	(mln,iyysiz,maxcpu)	寄与エネルギー (緑) の二乗の総和
colb2	R*8	(mln,iyysiz,maxcpu)	寄与エネルギー (青) の二乗の総和
ss	R*8	(mln,iyysiz,maxcpu)	ウエイトの総和
ncolor	I*4	(mln,iyysiz,maxcpu)	ピクセルに積算された光線の数

備考

モジュール記述シート

プログラム名	RTMC
モジュール名	DCARD
モジュールタイプ	common

機能

シナリオデータの情報

被呼出しモジュール	colset,input,mkdata,mkscrn,nextit,ray,readd, rgbout,rgbset,sourc1,sourc2,source,iblock

common/dcard/cname(nkd)

character cname*5

common/dcard1/xv,yv,zv

common/dcard2/xf,yf,zf,ixsize,iysize

common/dcard3/iscal,m0,n0

common/dcard4/xs(maxso),ys(maxso),zs(maxso)

common/dcard4/sor(maxso),sog(maxso),sob(maxso),nsoc,light(maxso)

common/dcard5/ref1(maxnc),ref2(maxnc),ref3(maxnc)

common/dcard5/dif1(maxnc),dif2(maxnc),dif3(maxnc)

common/dcard5/tra1(maxnc),tra2(maxnc),tra3(maxnc)

common/dcard5/refr(maxnc),nspe(maxnc)

common/dcard5/nmat,lmat(maxnc)

common/dcard6/cimp(maxnc),nimp

common/dcard7/xbara,ybara,zbara,rada,nflg_c8

common/dcard8/nnps

変数	型	配列	備考
cname	A*5	(nkd)	データカードのニーモニック
xv	R*8		視点の X 座標
yv	R*8		視点の Y 座標
zv	R*8		視点の Z 座標
xf	R*8		スクリーンの中心の X 座標
yf	R*8		スクリーンの中心の X 座標
zf	R*8		スクリーンの中心の X 座標
ixsize	R*8		スクリーンの横方向の大きさ
iysize	R*8		スクリーンの縦方向の大きさ
iscal	R*8		スクリーンのスケール値
	(続く)		

モジュール記述シート

プログラム名	RTMC
モジュール名	DCARD
モジュールタイプ	common

変数	型	配列	備考
	(続き)		
m0	R*8		スクリーンの横方向スケール値
n0	R*8		スクリーンの縦方向スケール値
xs	R*8	(maxso)	平行光線、円の半径
ys	R*8	(maxso)	平行光線、方位角
zs	R*8	(maxso)	平行光線、天頂角
sor	R*8	(maxso)	光源の赤成分の初期値
sog	R*8	(maxso)	光源の緑成分の初期値
sob	R*8	(maxso)	光源の青成分の初期値
nsoc	I*4		光線の数
light	I*4	(maxso)	光線の種類
ref1	R*8	(maxnc)	セルの赤成分の鏡面反射率
ref2	R*8	(maxnc)	セルの緑成分の鏡面反射率
ref3	R*8	(maxnc)	セルの青成分の鏡面反射率
dif1	R*8	(maxnc)	セルの赤成分の拡散反射率
dif2	R*8	(maxnc)	セルの緑成分の拡散反射率
dif3	R*8	(maxnc)	セルの青成分の拡散反射率
tra1	R*8	(maxnc)	セルの赤成分の透過率
tra2	R*8	(maxnc)	セルの緑成分の透過率
tra3	R*8	(maxnc)	セルの青成分の透過率
ref	R*8	(maxnc)	セルの屈折率
nmat	I*4		セル属性データの総数
lmat	I*4	(maxnc)	セル属性データに対応する番号
cimp	R*8	(maxnc)	ダミー
nimp	I*4		ダミー
	(続く)		

モジュール記述シート

プログラム名	RTMC
モジュール名	DCARD
モジュールタイプ	common

変数	型	配列	備考
	（続き）		
xbar	R*8		全セルを包み込むバウンダリ・ボリュームのＸ座標
ybar	R*8		全セルを包み込むバウンダリ・ボリュームのＹ座標
zbar	R*8		全セルを包み込むバウンダリ・ボリュームのＺ座標
rada	R*8		全セルを包み込むバウンダリ・ボリュームの半径
nflg_c8	R*8		全セルを包み込むバウンダリ・ボリュームの有無を示すフラグ
nnps	I*4		ヒストリー数

備考

モジュール記述シート

プログラム名	**RTMC**
モジュール名	**GEOM**
モジュールタイプ	**common**

機能

サーフェイスデータを格納する。

被呼出しモジュール	**chkcel,cross,cross2,input,** **surfcx,surfcy,surfcz,surfgq,surfkx,surfky,surfkz,** **surfp,surfpx,surfpy,surfpz,surfs,surfso,surfsq,** **surfsx,surfsy,surfsz,surftx,surfty,surftz,trace**

```
COMMON/GEOM/A$(IDIM),B$(IDIM),C$(IDIM),D$(IDIM),E$(IDIM)
COMMON/GEOM/F$(IDIM),G$(IDIM),H$(IDIM),J$(IDIM),K$(IDIM),R$(IDIM)
COMMON/GEOM/XBAR$(IDIM),YBAR$(IDIM),ZBAR$(IDIM),MARK$(IDIM)
REAL*8 H$,J$,K$
CHARACTER*4 MARK$
```

備考

サーフェイスの方程式の係数が格納されている。

モジュール記述シート

プログラム名	RTMC
モジュール名	GEOMGT
モジュールタイプ	common

機能

サーフェイスデータ（GENERAL TORUS）のデータを格納する。

被呼出しモジュール	chkcel,reads,surfgt

```
      COMMON/GEOMGT/
     + G01(IDIM2),G02(IDIM2),G03(IDIM2),G04(IDIM2),G05(IDIM2),
     + G06(IDIM2),G07(IDIM2),G08(IDIM2),G09(IDIM2),G10(IDIM2),
     + G11(IDIM2),G12(IDIM2),G13(IDIM2),G14(IDIM2),G15(IDIM2),
     + G16(IDIM2),G17(IDIM2),G18(IDIM2),G19(IDIM2),G20(IDIM2),
     + G21(IDIM2),G22(IDIM2),G23(IDIM2),G24(IDIM2),G25(IDIM2),
     + G26(IDIM2),G27(IDIM2),G28(IDIM2),G29(IDIM2),G30(IDIM2),
     + G31(IDIM2),G32(IDIM2),G33(IDIM2),G34(IDIM2),G35(IDIM2)
      COMMON/GEOMG2/
     + GX0(IDIM2),GY0(IDIM2),GZ0(IDIM2),GAA(IDIM2),GBB(IDIM2),
     + GCC(IDIM2),NUMGT
      REAL*8 G01,G02,G03,G04,G05,G06,G07,G08,G09,G10,
     * G11,G12,G13,G14,G15,G16,G17,G18,G19,G20,
     * G21,G22,G23,G24,G25,G26,G27,G28,G29,G30,
     * G31,G32,G33,G34,G35,
     * GX0,GY0,GZ0,GAA,GBB,GCC
```

備考

モジュール記述シート

プログラム名	RTMC
モジュール名	GRSIZE
モジュールタイプ	common

機能

スクリーン情報を格納する。

被呼出しモジュール	colset,input,mkscrn,rgbout,rgbset

common/grsize/ixsiz,iysiz,xlen,ylen

変数	型	配列	備考
ixsiz	I*4		スクリーンの横方向のピクセル数
iysiz	I*4		スクリーンの縦方向のピクセル数
xlen	R*8		スクリーンの横方向の大きさ
ylen	R*8		スクリーンの縦方向の大きさ

備考

モジュール記述シート

プログラム名	RTMC
モジュール名	ICOL
モジュールタイプ	common

機能

色情報を格納する。

被呼出しモジュール	colset,priprc,rgbout,rgbset

COMMON/ICOL/ICOLB1(MLN,IYYSIZ),ICOLG1(MLN,IYYSIZ)
COMMON/ICOL/ICOLR1(MLN,IYYSIZ),IRGB1(MLN,IYYSIZ)
COMMON/ICOL/ICOLB2(MLN,IYYSIZ),ICOLG2(MLN,IYYSIZ)
COMMON/ICOL/ICOLR2(MLN,IYYSIZ),IRGB2(MLN,IYYSIZ)
COMMON/ICOL/ICOLB3(MLN,IYYSIZ),ICOLG3(MLN,IYYSIZ)
COMMON/ICOL/ICOLR3(MLN,IYYSIZ),IRGB3(MLN,IYYSIZ)

変数	型	配列	備考
icolr1	I*4	(mln,iyysiz)	色情報 (赤)
icolg1	I*4	(mln,iyysiz)	色情報 (緑)
icolb1	I*4	(mln,iyysiz)	色情報 (青)
irgb1	I*4	(mln,iyysiz)	色情報 (ミックス)
icolr2	I*4	(mln,iyysiz)	色情報 (赤)
icolg2	I*4	(mln,iyysiz)	色情報 (緑)
icolb2	I*4	(mln,iyysiz)	色情報 (青)
irgb2	I*4	(mln,iyysiz)	色情報 (ミックス)
icolr3	I*4	(mln,iyysiz)	色情報 (赤)
icolg3	I*4	(mln,iyysiz)	色情報 (緑)
icolb3	I*4	(mln,iyysiz)	色情報 (青)
irgb3	I*4	(mln,iyysiz)	色情報 (ミックス)

備考

モジュール記述シート

プログラム名	**RTMC**
モジュール名	**ITEMC**
モジュールタイプ	**common**

機能

カードデータから読み込まれた１データを格納する。

被呼出しモジュール	**input,items,mkcel,mkdata,nextit,readc**

character hitm*80

common/itemc/ritm

common/itemc/nitm,iitm,irc

common/itemc/hitm

変数	型	配列	備考
ritm	**I*4**		実数データ
nitm	**I*4**		データの長さ
iitm	**I*4**		整数データ
irc	**I*4**		次に読み込む先頭カラム数
hitm	**A*80**		文字型データ

備考

モジュール記述シート

プログラム名	RTMC
モジュール名	SAVE
モジュールタイプ	common

機能

光線の各粒子の情報を一時的に保存するために使用。

被呼出しモジュール	cross,cross2,ray

```
COMMON/SAVE/XSAVE(NPART),YSAVE(NPART),ZSAVE(NPART)
COMMON/SAVE/XNSAVE(NPART),YNSAVE(NPART),ZNSAVE(NPART)
COMMON/SAVE/DSAVE(NPART),IDSAVE(NPART), IZSAVE(NPART)
```

変数	型	配列	備考
xsave	R*8	(npart)	save データ、交点のＸ座標
ysave	R*8	(npart)	save データ、交点のＹ座標
zsave	R*8	(npart)	save データ、交点のＺ座標
xnsave	R*8	(npart)	save データ、法線ベクトルのＵ成分
ynsave	R*8	(npart)	save データ、法線ベクトルのＶ成分
znsave	R*8	(npart)	save データ、法線ベクトルのＷ成分
dsave	R*8	(npart)	save データ、距離
idsave	R*8	(npart)	save データ、サーフェイス番号
izsave	R*8	(npart)	save データ、セル番号

備考

並列処理の場合、ローカルコモン

モジュール記述シート

プログラム名	RTMC
モジュール名	SCSURF
モジュールタイプ	common

機能

スクリーン情報を格納する。

被呼出しモジュール	colset,mkscrn,ray

common/scsurf/aaaa,bbbb,cccc,dddd,xs0,ys0,zs0

common/scsurf/xs1,ys1,zs1,xs2,ys2,zs2

変数	型	配列	備考
aaaa	R*8		スクリーン面の方程式の係数
bbbb	R*8		スクリーン面の方程式の係数
cccc	R*8		スクリーン面の方程式の係数
dddd	R*8		スクリーン面の方程式の係数
xs0	R*8		スクリーンの端点、X 座標
ys0	R*8		スクリーンの端点、Y 座標
zs0	R*8		スクリーンの端点、Z 座標
xs1	R*8		スクリーンの端点、X 座標
ys1	R*8		スクリーンの端点、Y 座標
zs1	R*8		スクリーンの端点、Z 座標
xs2	R*8		スクリーンの端点、X 座標
ys2	R*8		スクリーンの端点、Y 座標
zs2	R*8		スクリーンの端点、Z 座標

備考

モジュール記述シート

プログラム名	**RTMC**
モジュール名	**SNAME**
モジュールタイプ	**common**

機能

面方程式のニーモニック。

被呼出しモジュール	**chkcel,cross,cross2,trace,iblock**

character markv*4

common/sname/markv(27)

変数	型	配列	備考
markv	**A*4**	**(27)**	面方程式のニーモニック。

備考

markv はブロックデータで定義

モジュール記述シート

プログラム名	**RTMC**
モジュール名	**SURF**
モジュールタイプ	**common**

機能

サーフェイスの総数、サーフェイス番号を格納

被呼出しモジュール	**chekcs,reads,namchg**

common/surf/msurf,lsur(maxs)

変数	型	配列	備考
msurf	**I*4**		サーフェイス総数
lsur	**I*4**	**(maxs)**	サーフェイス番号

備考

lsurfは **namchg** で通し番号に変換される。

各サーフェイスの情報はコモンＧＥＯＭ及びＧＥＯＭＧＴに格納される。

モジュール記述シート

プログラム名	RTMC
モジュール名	PARTCL
モジュールタイプ	common

機能

光線の各粒子の情報を格納する。

被呼出しモジュール	cross,cross2,ray,source

common/partcl/x0(npart),y0(npart),z0(npart)
common/partcl/u(npart), v(npart), w(npart)
common/partcl/eng1(npart),eng2(npart),eng3(npart)
common/partcl/wt(npart),refrs(npart)
common/partcl/id(npart),izz(npart),nflgs(npart)

変数	型	配列	備考
x0	R*8	(npart)	現在位置のX座標
y0	R*8	(npart)	現在位置のY座標
z0	R*8	(npart)	現在位置のz座標
u	R*8	(npart)	現在の方向余弦のU成分
v	R*8	(npart)	現在の方向余弦のV成分
w	R*8	(npart)	現在の方向余弦のW成分
eng1	R*8	(npart)	現在のエネルギー（赤）
eng2	R*8	(npart)	現在のエネルギー（緑）
eng3	R*8	(npart)	現在のエネルギー（青）
wt	R*8	(npart)	現在のウエイト
refrs	R*8	(npart)	屈折率
id	I*4	(npart)	現在のサーフェイス番号
izz	I*4	(npart)	現在のセル番号
nflgs	I*4	(npart)	減衰させるかどうかを示すフラグ

備考

並列処理の場合、ローカルコモン

モジュール記述シート

プログラム名	RTMC
モジュール名	PARTC2
モジュールタイプ	common

機能

光線の各粒子の情報を格納する。

被呼出しモジュール	colset,cross,cross2,ray

COMMON/PARTC2/X(NPART) , Y(NPART) , Z(NPART)
COMMON/PARTC2/XN(NPART), YN(NPART), ZN(NPART)
COMMON/PARTC2/XR(NPART), YR(NPART), ZR(NPART)

変数	型	配列	備考
x	R*8	(npart)	衝突点の X 座標
y	R*8	(npart)	衝突点の Y 座標
z	R*8	(npart)	衝突点の Z 座標
xn	R*8	(npart)	衝突点での法線ベクトル
yn	R*8	(npart)	衝突点での法線ベクトル
zn	R*8	(npart)	衝突点での法線ベクトル
xr	R*8	(npart)	鏡面反射の方向余弦
yr	R*8	(npart)	鏡面反射の方向余弦
zr	R*8	(npart)	鏡面反射の方向余弦

備考

並列処理の場合、ローカルコモン

モジュール記述シート

プログラム名	RTMC
モジュール名	PARAM
モジュールタイプ	parameter

機能

パラメータを設定する

被呼出しモジュール	全ルーチン

```
parameter(iic=12,ioc=80,iis=11,ios=80,iid=14,ioe=80)
parameter(iiib=13,iob=80)
parameter(IDIM=5300,IDIM2=300)
parameter(MLN=1024,IYYSIZ=1152)
parameter(maxc=3000)
parameter(maxs=5300,maxnc=40)
parameter(maxso=10)
parameter(insurf=500)
parameter(nkd=11)
parameter(maxbv=20)
parameter(cdiv1=10.0,cdiv2=0.01,cdiv3=1.0)
parameter(dnorm=1.0D+05)
parameter(wlow=0.15,wavg=0.5)
parameter(mstr=10)
parameter(pi= 3.14159265)
parameter(nrr=153600)
parameter(npart=2500)
parameter(maxcpu=4)
```

備考