GPUPPUR:汎用高速 3D グラフィックスライブラリ

松本知大
富山県立大学 工学部電子情報工学科

概要

三次元情報(3D)を映像化する技術は主にレイトレーシング法とラスタライズ法に分けられる。ラスタライズ法はゲームなどのリアルタイムに 3D を表示する分野で主に使われ、レイトレーシング法は映画など時間を掛けて綺麗な映像を作る分野で使われている。最近では多くの PC にラスタライズ法で高速にレンダリングを行うための専用ハードウェアである GPU が搭載されている。また、ゲームにおけるリアリティーを向上させるために物理シミュレーションが行われるようになっているが、この処理を専門に行うハードウェアである PPU が一般向けに販売されている。

GPUPPUR は 3D グラフィックスレンダリングライブラリとして開発している。GPUPPUR はレイトレーシングを行なう GPUPPURay とラスタライズを行なう GPUPPURas から構成されている。

レイトレーシング法では、視点から各画素に対応する光線をキャストし、光線と物体が交差した位置からの光を計算して画素の色を求める。GPUPPURay では、光線と物体の交差を求める処理を PPU で行い、次の画素の色を求める処理は GPU で行なうことによって高速に処理することを目指す。

ラスタライズ法では、三角形を一つ一つ 2 次元画像へと変換するが、視界に入っている物体のみを効率良く探してラスタライズすることによって、見えない物体を処理する無駄を省くことができる。レイトレーシングでの視点からの光線と物体との交差判定結果によって、どの物体が視界に入っているかを知ることができる。GPUPPURas では、GPUPPURay と同様に PPU を利用して光線と物体の交差判定を行い、交差判定結果から可視物体のみをラスタライズすることによって効率良く処理を行う。

ラスタライズ法は図形の数 n に対して計算量は O(n)だが、レイトレーシング法では O(log(n))である(*1)。また、ラスタライズ法では画素数 m に対して計算量は O(log(m))だがレイトレーシング法では O(m)である(*2)。このことから、図形の量や画素数に応じてレイトレーシング法とラスタライズ法を使い分けることができれば常に高速にレンダリングが行えると考えられる。GPUPPUR ではこの機能の実現も目指す。

1 背景

コンピュータで 3 次元情報(3D)を 2 次元の画像で表現する方法は様々である。現在のパーソナルコンピュータでリアルタイムに 3D を表示するときにはラスタライズ法が使われている。一方で、時間が掛かってでも綺麗な画像が欲しいというときは、レイトレーシング法やその改良が使われている。ラスタライズ法は物体を三角形のみで構成し、その三角形を一つ一つ2次元画像に変換することによって目的の 2 次元画像を生成する。レイトレーシング法は、視点からそれぞれの画素を通る光線を投げて(キャスト)、その光線がどの図形と交差するか、交差した点からどのような光が反射しているかなどを計算し、画素にどのような光が入ってくるかを調べる。

表 1 にラスタライズ法とレイトレーシング法の比較を示す。ラスタライズ法は比較的高速に処理がしやすく、一般消費者向けの安価な専用ハードウェア(Graphics Processing Unit, 以下 GPU)上で複雑な物体をリアルタイムに処理することができる。そのため、3D を使ったほとんどのコンピュータゲームではラスタライズ法が使われている。一方でレイトレーシングを専用に行なうハードウェアを開発することは難しく、まだ一般消費者向けの安価な製品は売られていない。現在のところレイト

レーシング法でコンピュータゲームの 3 次元図形を十分な速さで処理することは難しい。そのためには高価なコンピュータが必要となる。

2 目的

GPUPPUR は 3D グラフィックスのレンダリングエンジンであり、プログラムにリンクして使用するライブラリとして開発している。

GPUPPUR の目標を以下に示す。

- ライブラリ利用者の創造性を発揮することができ、独創的な映像を生み出すことができる。
- カスタマイズや特殊な機能を使わない限りは、様々なプラットフォームへ簡単に移植すること、様々なハードウェアに対応することを可能にする。
- カスタマイズすることによって、特定のハードウェア、OS 上でしか動かすことができないが、ハードウェアの能力をフルに発揮できるアプリケーションの開発を可能にする。

GPUPPUR にはレイトレーシングを行う GPUPPURay とラスタライズを行う GPUPPURas を実装する予定である。

また、GPUPPUR をオープンソースで公開し、誰もが自由に GPUPPUR を使ったり改変できるようにする予定である。

表 1 レイトレーシング法とラスタライズ法の比較

	レイトレーシング法	***ラスタライズ法***
扱う図形	3D グラフィクスで扱える図形全般	三角形のみ
描画時間	遅い	速い
専用ハードウェア	開発中、一般には入手困難	安価に入手可
2 次元画像の質	高品質	低品質

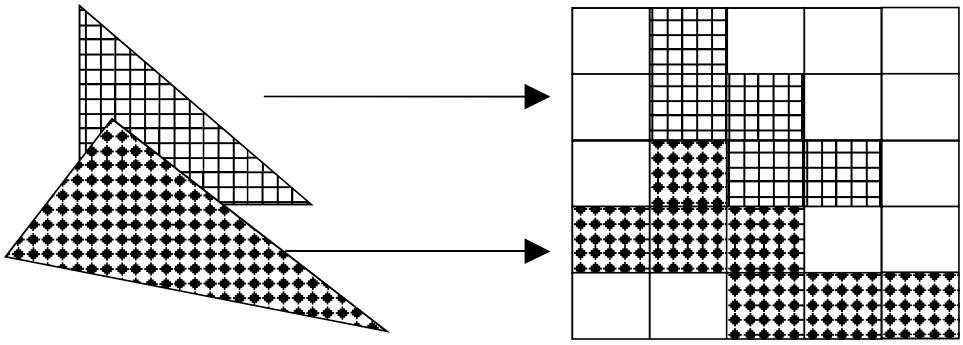

図1 ラスタライズ法。それぞれの三角形について、2次元画像へ変換する処理を行う。

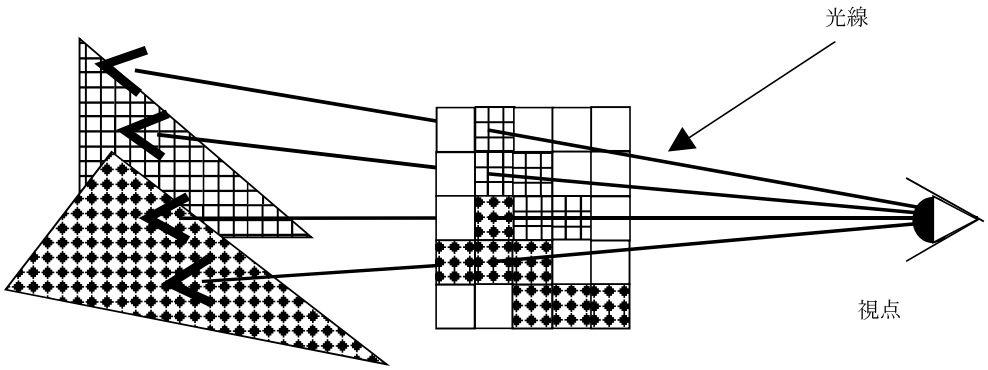

図2 レイトレーシング法。視点から光線をキャストし、それぞれの光線が物体と交差するか、交差する点での光の様子などを計算する。

3 要素技術

図1、図2にそれぞれラスタライズ法、レイトレーシング法を簡単な図に示す。

ラスタライズ法の処理を行なうGPUの大きな特徴の一つは、三角形の頂点に関する値(頂点の座標、色など)を処理するプロセッサ(Vertex Shader, 以下VS)や画素単位の処理を行うプロセッサ(Pixel Shader, 以下PS)が複数搭載されていることである。VSやPSで行われる処理は並列化できるため、大量の三角形を描画するときでも複数のVSとPSが並列に動作することによって高速に処理される。さらに最近のGPUでは、VSとPSでの処理をプログラミングすることができるため、ある程度自由に処理を制御することができる。だが、1つの三角形を処理しているときに他の三角形の情報を参照することができないため、影や反射、屈折等の周囲の図形からの影響を考慮した処理を行なうのは難しい。一方でレイトレーシング法では、計算時にすべて

の図形や光源の情報を持っている。そのため、空間内にある全ての図形の情報を使って、光の伝わり方を正確に計算することができるため、影や反射、屈折を簡単に処理できる。

ラスタライズ法では、求める画像の画素数が増えても処理時間はあまり増加しないが、レイトレーシング法では、処理時間は画素数に比例する。一方で、ラスタライズ法は画像に描画される三角形の数に比例した処理時間が必要となるが、レイトレーシング法では三角形の数が増えても処理時間はあまり増加しない。ラスタライズ法とレイトレーシング法での、画素数が一定の場合に三角形の数が増えたときの処理時間の増加の様子を図3に示す。このことから、画素数があまり多くないが、かなり複雑な3D形状を描画したいとき、レイトレーシング法のほうが高速に計算できると言える。また、三角形の数や画素数を考慮して、ラスタライズ法とレイトレーシング法を切り替えることができれば、三角形の数が激しく増減する場合でも常に高速に描画することができると考えられる。図4にラスタライズ法とレイトレーシング法を切り替えて処理する場合での、三角形の数と処理時間の関係を示す。

レイトレーシング法では、物体と光線との交差判定が処理時間の多くを占めるため、高速化するにあたって交差判定処理は重要になってくる。また、光線と物体の交差判定の結果から、どの物体がカメラから見えるかを知ることができる。この情報をラスタライズ法で利用すれば、見えない物体の描画処理を行う無駄を減らすことができる。

最近、ゲームの世界の中の物体を物理的に正しく動かせて現実感を向上させるために、物理シミュレーションを行うコンピュータゲームが増えている。ゲームに必要な物理シミュレーションに要求される負荷が今後増大することを考え、物理シミュレーション専用ハードウェア(Physical Processing Unit，以下PPU)が一般向けに発売されている。

物理シミュレーションでは、物体同士が衝突したことを検出し、それぞれの物体が重なることや貫通することがないように処理しなければならない。PPUでは物体同士の衝突検出を効率良く行えるように、空間分割を行いできるだけ近くにある物体同士の衝突判定だけを行えば済むように工夫している。この工夫を利用することと、専用に設計されたハードウェアによって、PPUでは高速に物体と光線との交点の情報を得られる。

4 GPUPPURについて

GPUPPUR(Graphics Purpose Unlimited Programmable Portable Utilized Renderer ”ぐぷっぱー”と発音する。)は3次元グラフィックスのレンダリングエンジンであり、3次元図形を扱うプログラムにリンクして使われるライブラリとして開発する。複雑な3D形状を高速に描画する機能等を提供することを目指す。また、汎用3DAPIとして移植性に優れながらも特定の目的用にカスタマイズでき、開発者の創造力をフルに発揮できるようなライブラリとすることを目指す。

GPUPPURの実際の3Dグラフィックス描画機能として、レイトレーシング法を使用するGPUPPURay(Graphics Processing Unit and Physical Processing Unit for Raytracingの略)とラスタライズ法を使用するGPUPPURas(Graphics Processing Unit and Physical Processing Unit for Rasterizingの略)を実装する。どちらの実装も従来のGPUとCPUだけで計算する場合より、複雑な3Dグラフィックスの高速な描画をGPUとPPUを活用することによって実現することを目指す。GPUとPPUが利用できる一般的なPC上で動作できるようにする。

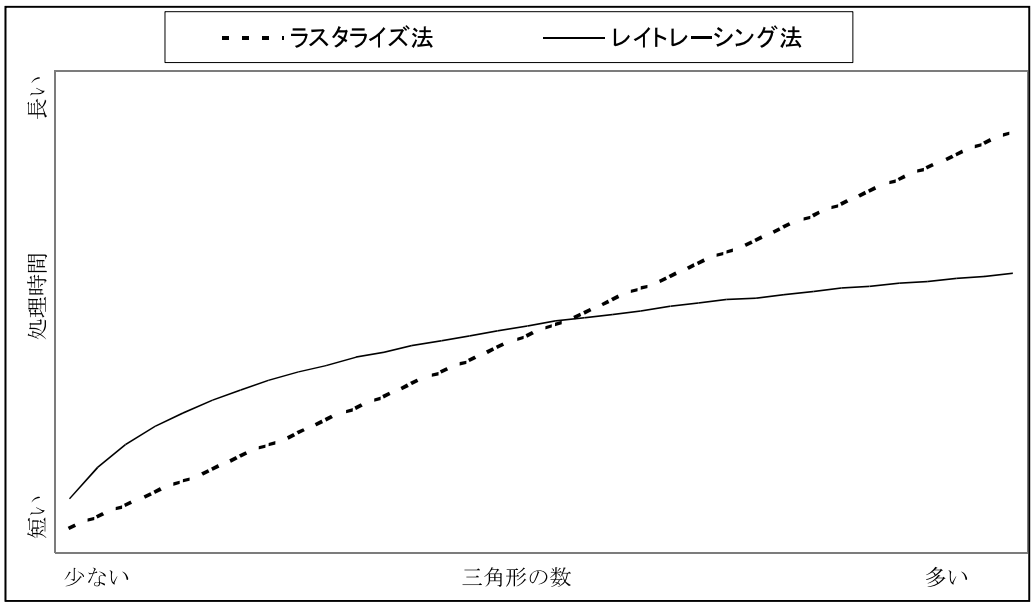

図 3　ラスタライズ法、レイトレーシング法での、三角形の数と処理時間の関係

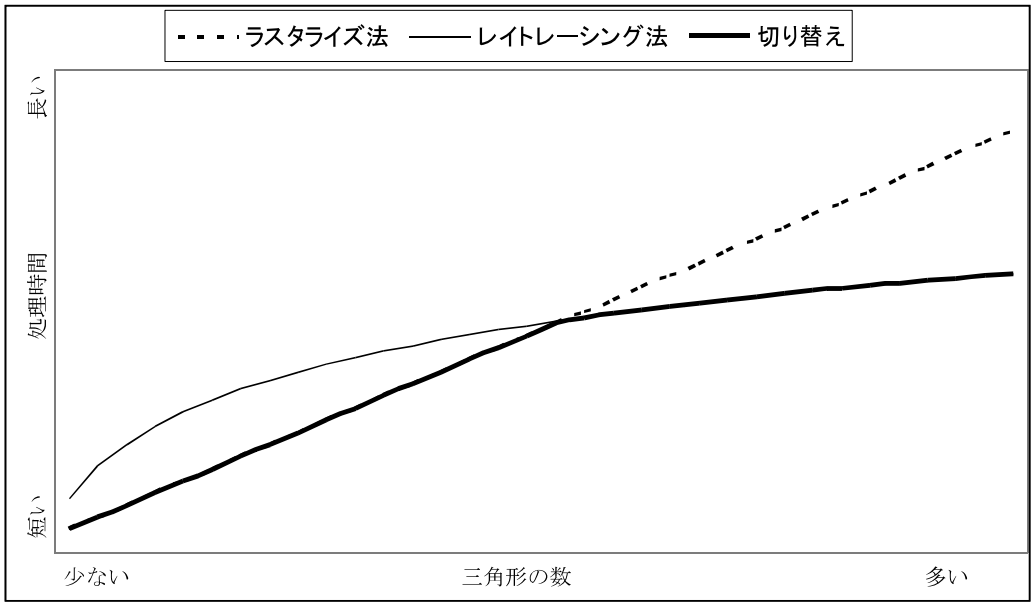

図 4　ラスタライズ法とレイトレーシング法の切り替えを行った場合の三角形の数と処理時間の関係

GPUPPURay と GPUPPURas は、PPU を PhysX の API を通して利用し、GPU を OpenGL か Direct3D を通して利用する。

GPUPPURay、GPUPPURas で一枚の画像を計算して表示するときの主な処理は以下のような流れになる。以下の処理の流れを図にしたものを図 5 に示す。

1、CPU は描画したい図形の情報を PPU へ送る。
2、CPU は求める画像の画素の数だけキャストする光線の情報を PPU へ送る。
3、PPU で光線と図形の交差判定を行う。
4、PPU での交差判定の結果(交差した図形の識別番号、レイトレーシングの場合は交点座標、交点での法線ベクトル等も)を CPU へ送る。

レイトレーシングを行う場合
5、CPU は交差判定の結果から交差した点での材質等を求め、交差した点での座標、法線ベクトル、材質等の色の計算に必要なデータを揃える。
6、CPU は 5 で揃えたデータを GPU へ送る。
7、GPU で送られてきた情報を元に色の計算を行い、結果を画面に表示する。

ラスタライズを行う場合
5、CPU は交差判定の結果を調べ、光線と交差した物体の描画命令を GPU へ送る。
6、GPU で、送られた描画命令を実行して、物体を表示する。

目的や条件によっては処理の流れが異なる場合がある。例えば、レイトレーシング法で影を表現したい場合、処理 4 の後に交点から光源方向への光線を、求める画像の画素の数だけ再び PPU へ送り処理 3、4 のような処理を行う。また、描画結果に対して Direct3D や OpenGL の API を使用して別のオブジェクトを上書きすることができるようにする。レイトレーシング法による出力結果に”色”に加えて”深度”も出力することによって Z buffer 法による陰面消去が可能になる。そうすることによって、レイトレーシング法で描画した後にラスタライズ法で別の物体を追加して描画することができる。

5 レイトレーシング法とラスタライズ法の組み合わせ

レイトレーシング法で効率的に処理できる場合とラスタライズ法のほうが効率的に処理できる場合がある。レイトレーシング法では球や円柱等を効率良くレンダリングできるが、ラスタライズ法ではそれらの図形を複数の三角形に分割して表示する必要があるためあまり効率が良くない。レイトレーシング法では、三角形で構成された図形を高速にレンダリングするために前処理を行なう。そのため、常に形が変化するように物体をアニメーションさせる場合に、画面を更新する度に前処理を行なう必要がでてくるために滑らかに形を変形させることが難しくなる。ラスタライズ法ではそのような前処理が必要無いために高速に処理することができる。

そこで、各物体をより適した方法でレンダリングしそれらを組み合わせる方法について説明する。
複数の物体を違う方法でレンダリングするときに、以下のことが問題となる。
(1)陰面消去
(2)影
(3)反射や屈折
(1)について、ラスタライズ法では Z buffer 法を使って陰面消去する方法が主に利用されている。Z buffer 法とは、画面の各ピクセルに対応した Z 値を格納したバッファを使った方法である。レンダリングされた三角形の各ピクセルの Z 値が Z buffer にある値よりも小さい場合に新しくそのピクセルの Z 値を格納し、そのピクセルの色を設定する。逆に Z buffer にある値よりも Z 値が大きい場合は色を設定しないことによって、画面に近いほうの三角形が遠いほうの三角形の手前に表示されることがなくなる。レイトレーシング法では、通常は画素の色のみを計算して画面へ出力するが、光線と物体の交点の Z 値も計算することによって、Z buffer 法を使った陰面消去をすることができる。

レイトレーシング法を使って影や反射、屈折を表現する場合、ラスタライズ法で処理したい図形もレイトレーシングの計算に含める方法が考えられるが、そのままでは計算時間が大幅に増えてしまう。影や反射、屈折によって映される映像が画面のリアリティーにあまり多く影響しない場合は、ラスタライズ法でレンダリングする物体を含めないか、単純化して高速にレイトレーシングできるようにした物体を使って計算することによって、それらしく見える結果を得ることができるだろうと考えられる。

ラスタライズ法では、反射の表現を近似するために環境マッピングという手法が使われている。物体の周囲の映像を画像にして、物体の表面に周囲の映像が反射されて映って見えるように物体に画像をテクスチャとして貼り付ける方法がある。レイトレーシング法でも同様の手法を使うことによって、反射を擬似的に表現することが可能である。

6 GPUPPURの現状

GPUPPURay、GPUPPURasで、(いくつか条件はあるがほぼ)任意の三角形から構成される図形をレンダリングし、位置や方法を自由に動かしてアニメーションさせることができる。また、カメラの位置や向き等も変えることができる。

GPUPPURayやGPUPPURasはGPUとPPUを活用してレンダリングを行なう予定ではあるが、現在はGPUだけしか活用できていない。PPUは今のところ決まったAPIの関数を通してしかPPUの機能にアクセスできないのだが、光線と物体の交差判定をPPUで行なうAPIがまだ公開されていないためにPPUを活用することができない状況である。

7 GPUPPURのこれから

GPUPPURでさらに多くの表現が可能になるように以下の機能を追加する予定である。

・テクスチャマッピング

・GPUPPURayで球体、平面、箱等の三角形以外の図形をレンダリング

・GPUPPURayで影、反射、屈折の表現

また、いくつかの高速化技術を使ったり、処理内容を見直す等して、さらにGPUPPURが高速にレンダリングできるようにする。

参考文献

(*1) Matt Pharr and Greg Humphreys: *Physically Based Rendering - From Theory To Implementation* pp.8: Morgan Kaufmann Pub 2004年

(*2) J¨org Schmittler: SaarCOR A Hardware-Architecture for Realtime Ray Tracing: Report of Computer Graphics Group Saarland University Saarbr¨ucken, Germany:
URL http://graphics.cs.uni-sb.de/~jofis/Schmittler-SaarCOR_Phd-HighRes-preliminary.pdf

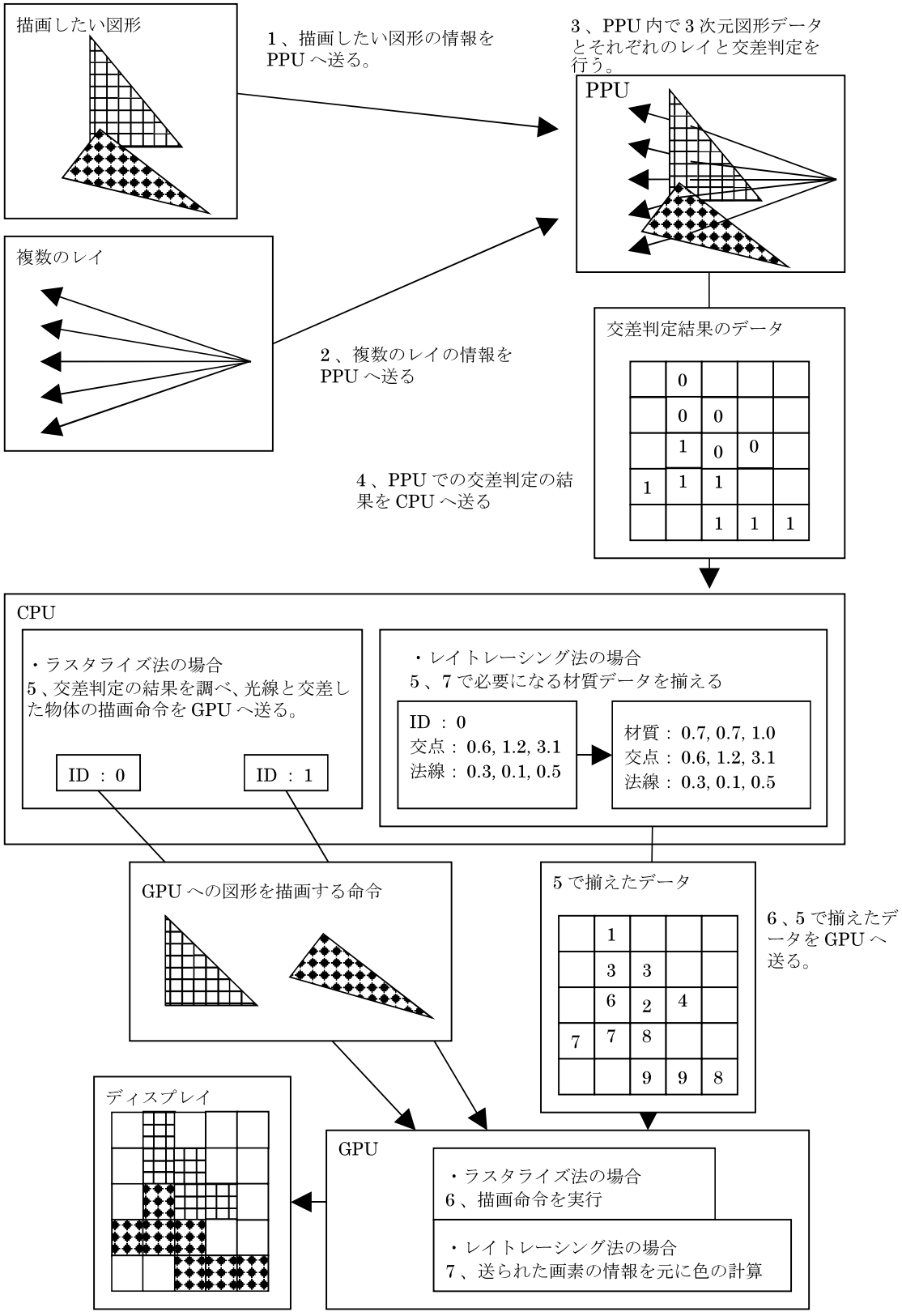

図 5　GPUPPUR の仕組み