2008－MPS－68 (2)
2008／3／4

リアルタイムQueryballの開発に向けて

勅使河原佑美* 来見真理* 石川千里**
高田雅美** 城和貴**
*奈良女子大学理学部情報科学科
**奈良女子大学院人間文化研究科情報科学専攻

3次元空間内のオブジェクトを理解するための機能の1つとしてQueryballがある．Querybllはオブジェクトに半透明の球体を重ね合わせることにより，オブジェクトと球体が重なる部分に対して問合せることができ，オブジェクトの表示方法を変えたり，様々な方向からオブジェクトを視認したりできる．しかし，現段階のQueryballのボリュームレンダリングでの表示速度はオブジェクトデータのサイズが大きくなるほど遅くなり，実用には向かない．そこで，Querybllをリアルタイムで実行するために，ボリュームレンダリング部分にGPGPUを適用することによって，高速化を図る．

Toward the Development of Real-time Queryball

Yumi Teshigawara*, Mari kurumi*, Chisato Ishikawa**,
Masami Takata**, Kazuki Joe**

*Department of Information and Computer Sciences, Nara Women's University
**Graduate School of Humanities and Sciences, Nara Women's University

Queryball is a tool to understand objects in three-dimensional space. By overlapping a translucent globe with 3D object, it can quiry to the part where the object and the globe come in succession. It can change the display method with visually checking according to various directions. However, the display speed the volume rendering of Queryball to the present stage slows the large size of the object data, and is not for practical use. Therefore, performance improvement by applying GPGPU to the volume rendering part in real time is necessary.

1. はじめに

現代の情報社会には，多量のデータが累積している．この多量のデータを解析する有効な手段の1つとして，データの3次元可視化がある．データを3次元可視化することにより，データを分かりやすく直感的に捕らえることができ，また新たなデータ間の関連を理解することができる．この3次元可視化に対し，仮想空間とデータベースを関連付け，直接問合せができるようなシステムがあれば，データの詳細をよりわかりやすく捕らえることができる．

そのためのシステムとして，渡辺が開発したQueryball[1][2]がある．Querybllは半透明の球体を操作することにより，直感的にデータオブジェクトと球体の重なった部分に対して問合せすることができる．Querybllは，データの可視化方法の1つであるボリュームレンダリングの表示速度が遅いという問題点がある．そこで本研究では，Querybllのボリュームレンダリングの計算部分をGPGPUにより高速化する．

2. Queryball

2.1 概要

Querybllは，VTK(Visual Toolkit)[3]を用いて開発されたシステムである．Querybllは半透明の球体を操作することができ，データに球体を重ね合わせることにより，データに対し直接問合せることができる．

Querybllは，領域条件，探索条件，探索条件に該当する部分と該当しない部分の表示方法でフィルタが定義されている．図1は，縦に4個，横に3，奥行に4個ボクセルが並んでいる．各ボクセルには属性valがあるとする．図1は，領域条件は球体内部，探索条件はvalの値が10以上，探索条件に該当する表示方法は赤色で表示し，探索条件に該当しない部分は，透明で外枠のみ

フレームで表示すると定義したものである.

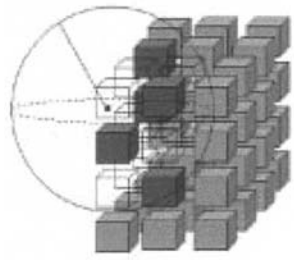

図 1：Queryball の例

2.2 ボリュームレンダリング

Queryball は VTK で構成されているため，ボリュームレンダリングのパイプラインは図 2 のようになっている.

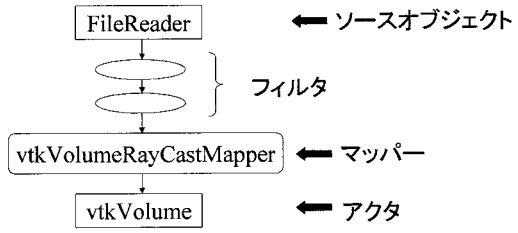

図 2：ボリュームレダンリングのパイプライン

まずファイルの読み込みやデータセットを作成するソースオブジェクトにより，可視化したいデータを読み込み，データセットを作成する．次に作成されたデータセットをいくつかのフィルタに通す.

次にデータセットを実際に表示できる形式に変える．今回はボリュームレンダリングの中でもレイ・キャスティング法を用いるので，vtkVolumeRayCastMapper というマッパーを用いる．データセットにレイ・キャスティング法を適用し，3 次元空間で描画されるオブジェクトにマッピングする．最後に vtkVolume というボリュームレデータ用のアクタに変更し，レンダラに登録する.

ボリュームレンダリングの計算時間は可視化するデータのサイズが大きいほど時間がかかり，現時点では 100×100×100 のデータを通常の PC でボリュームレンダリングするにあたり，1 秒間に 4 枚しか表示することができない.

3. GPGPU

GPGPU[4]は，「グラフィックスハードウェアを用いた汎用計算」のことである．本来グラフィックスを生成するための GPU を，意図して他の計算用途に使ことである.

3.1 GPGPU を行うための環境

GPGPU ではグラフィックスカード，プログラムのコンパイラ，グラフィックス API の環境を整える必要がある．3.3 の実験ではグラフィックスカードを数種類用いて比較をする．コンパイラには Queryball の GUI に使用されている VisualStudio2005 を用いる．グラフィックス API はＶＴＫのローベル API でもある OpenGL を用いる.

GPGPU ではシェーダを使用して，プログラマブルパイプラインをプログラムする．3.2 の実験ではシェーディング言語に Cg を用いる．Cg は NVIDIA によって開発され，C 言語をベースとした文法を持つ．また，Queryball への GPGU の適用の際は，glsl(OpenGL Shading Language)を使用する．glsl も C 言語をベースとした高レベルシェーディング言語で OpenGL との親和性が高く，間のインターフェースが分かりやすい.

3.2 GPU のスペックによる比較実験

GPGPU によりボリュームレンダリングの高速化が可能であるかを調べるため，スペックの異なる 6 種類の GPU を用いた比較実験を行う.

プログラムには，Peter Triers Blog[8]に掲載されているプログラムを使用する．このプログラムは，レイ・キャスティング法によるボリュームレンダリングで，GPGPU により高速化してある．データ量を 512*512*512 に変更し，以下の 6 種類の GPU を用いて実験を行う.

No.	GPU
1	Geforce8800GTX メモリサイズ：768 MB VRAM のバンド幅：86.4GB/s
2	QuadroFX4500 メモリサイズ：512 MB VRAM のバンド幅：33.6GB/s
3	QuadroFX1400 メモリサイズ：128 MB VRAM のバンド幅：19.2 GB/s
4	GeForce7300GS

	メモリサイズ：512 MB VRAM のバンド幅：6.5 GB/s
5	GeForce7200GS メモリサイズ：256 MB VRAM のバンド幅：6.4 GB/s
6	GeForce6200 メモリサイズ：128MB VRAM のバンド幅：2.1 GB/s

結果は図 3 のようになった．

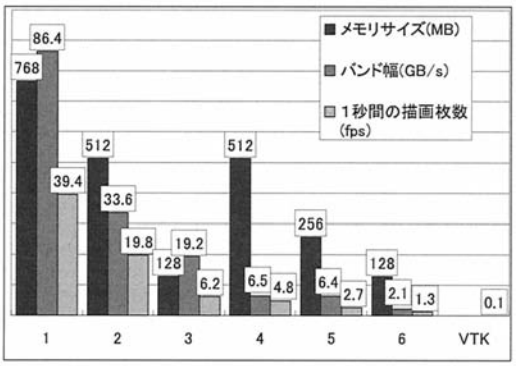

図 3：GPU のスペックの違いによる計算処理性能比較実験

GPU のメモリサイズと VRAM のバンド幅が大きいほど 1 秒間の描画枚数が多いことが分かる。また CPU でボリュームレンダリングを行っている VTK での 1 秒間の描画枚数は 0.1 となった．実験に使用した GPU のうち最もスペックの良いものでは，約 400 倍の描画速度が可能である．よって GPGPU によりボリュームレンダリングの高速化は可能であるといえる．

4. GPGPU による Queryball の高速化

4.1 Queryball への GPGPU の適用

Queryball の高速化の方法は 2 種類考えられる．

1 つ目は，ボリュームレンダリングの計算から描画までを，VTK を使用せず 3.2 の実験で用いたサンプルプログラムを用いて実行する方法である．

この方法では CPU から GPU へデータを 1 度転送するだけですむが，Queryball で使用できる多くの VTK の関数との互換性がなくなってしまうため困難である．

2 つ目は，VTK のボリュームレンダリングの計算部分に GPGPU を適用し，計算結果を取得し描画は VTK で行う方法である．この場合，CPU から GPU，GPU から CPU へのデータ転送が必要なため，1 つ目ほどの高速化は不可能であるが，他の VTK の関数との互換性の心配はないため，この方法で高速化する．

VTK のボリュームレンダリングの計算は vtkVolumeRayCastMapper.cxx というファイルの中で行われている．このクラス内部のボリュームレンダリングの計算部分に GPGPU を適用する．

ボリュームレンダリング計算途中にも VTK 関数がでてくるため，まず，

CPU の計算時間
　＞GPU の計算時間＋ CPU⇔GPU の転送時間

が成り立つ部分のみを GPU で行う．

4.2 GPGPU を用いた数値計算プログラム

GPGPU ではデータをテクスチャとしてビデオメモリに送り，GPU で計算させる．GPU で計算された結果は，通常はフレームバッファに出力され，描画ウィンドウに表示される．描画ウィンドウには出力せず，計算結果だけを得るための方法として，次の 2 つが考えられる．

1 つ目は，ダブルバッファを用いたオフスクリーンレンダリング法である．オフスクリーンで描画したデータから計算結果を得る．

2 つ目は，計算結果の出力先をフレームバッファから新たに用意したテクスチャに変更する方法である．このとき，FBO(Framebuffer Object)という方法を利用する．FBO は OpenGL でプラットフォームに依存せずオフスクリーンレンダリングを実現する機構である．従来のフレームバッファは，実際の表示ウィンドウと密接に関連付けられていたが，FBO は従来のフレームバッファと同等の論理バッファの組を，表示ウィンドウと別個に保持し、それに対してレンダリングすることが可能となっている．そのため，テクスチャへのレンダリングも可能である．

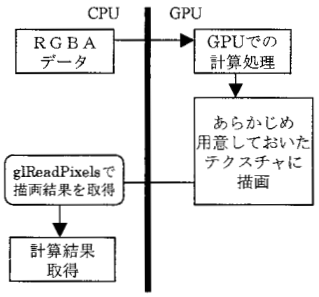

図 4：ボリュームレダンリングのパイプライン

そこで，GPU で数値計算をするプログラムを作成する．まず RGBA を取り扱うテクスチャを用意し，そこに計算に使用するオブジェクトデータを配置することで，データを GPU に転送する．又，計算に使用するその他の数値もテクスチャとして GPU に転送する．

そしてシェーダ言語を用い，GPU で計算を行う．そこで得た値を FBO により，あらかじめ用意しておいたテクスチャ R に描画し，OpenGL 関数の glReadPixels により計算結果を得る．このときＲＧＢＡの各値を同時に受け取ることができる．

5. おわりに

本研究の目的は，Queryball のボリュームレンダリングに GPGPU を適用することにより高速化することである．そのために，GPGPU どれだけの高速化が可能なのか GPU のスペックによる比較実験を行った．その結果，GPU のメモリと VRAM のバンド幅により 1 秒間に何枚描画可能なのか推定できた．また，CPU でボリュームレンダリングの計算を行う場合と比べて，約 20 倍の高速化が可能であった．これにより，GPGPU により，ボリュームレンダリングの高速化が期待できることも分かった．

また，Queryball のボリュームレンダリング部分への GPGPU の適用について，ボリュームレンダリング の計算から描画までを，実験に使用したサンプルプログラムを用いて行う方法では，他の VTK の関数との相互関係がなくなってしまう．そのため VTK プログラム内のボリュームレンダリング計算部分に GPGPU を適用し，得た数値を VTK プログラム内に読み込む方法をとる．

現在の手法より，より良い高速化を行うためには，VTK でのボリュームレンダリングの計算部分に含まれる VTK の関数部分も GPU で処理する必要がある．

また，ボリュームレンダリングだけでなく，ベクトルデータやポイントデータからの RGBA 値への変換も GPU で計算することが可能であると考えられる．それにより，ボリュームレンダリングを行うまでのデータ処理部分も高速化される．

最終的には，Queryball を本学の可視化工房[9]で実装したいと考えている．ステレオ表示による立体視を行うため，1 秒間に左右 30 枚ずつ合計 60 枚の描画を行いたい．また，入力インターフェースに Wii リモコンを使用したいと考えている．これにより，よりリアルにデータへ問合せることが可能となる．

参考文献

[1] 渡辺知恵美，増永良文，城和貴 ：Queryball: 没入型 VR システムのための対話的な問合せモデル，日本データベース学会 Letters, Vol.2, No.2,pp.25-28 (2003)

[2] Ai Ishida, Chiemi Watanabe and Kazuki Joe: A Query Description Model and its Implementation as an Interactive Query Tool for Visualization Systems, The 2004 International Conference on Parallel and Distributed Processing Techniques and Applications,

[3] The Visualization ToolKit http://public.kitware.com/VTK

[4]GPGPU http://www.gpgpu.org/

[5] GPU Programming http://www.geocities.jp/takashi_drive2005/gpuindex.htm

[6] NVIDIA http://www.nvidia.co.jp/page/home.html

[7] Sgi OpenGL Shading Language http:/www.sgi.com/support/custeducation/emea/courses/sgi/opengl_shading.html

[8]Peter Triers Blog GPU raycasting tutorial http://www.daimi.au.dk/~trier/?page_id=98

[9] 奈良女子大学理学部 現代 GP 可視化ＧＰとは http://hpcl.ics.nara-wu.ac.jp/vgp/p_purpose.html

謝辞

本研究遂行にあたって，様々な協力やアドバイスをいただ Queryball 考案者の渡辺知恵美講師に感謝する．