数理モデル化と問題解決 33－11
（2001．3．16）

DirectXを用いたNaraViewの高速化

柴山智子　城 和貴
zero1977@ics.nara-wu.ac.jp
奈良女子大学 理学部 情報科学科

概要

NaraViewは与えられたプログラムから必要な情報を抽出し、3次元空間上に視覚化する。その役割は人間の優れた視覚認識能力による判別を通しユーザー自身にプログラム構造を把握させることでプログラムの並列化を支援することである。本稿ではNaraViewの現状を踏まえた上で、DirectXを用いた再実装の試みについて述べる。

Improvement in the speed of NaraView by DirectX

Tomoko Shibayama　Kazuki Joe
Nara Women's University

Abstruct

NaraView is a tool which supprts parallelizations of given programs with the advantage of human's visual perception ability. This paper describes the re-implementation of NaraView with DirectX, after the comparison of OpenGL and DirectX.

1　はじめに

NaraView[2]はプログラム構造やデータ依存情報を3次元空間上に視覚的に表現する。このツールを利用することで、ユーザーは最適化手法の具体的な効果を直感的に理解し、より有効な並列化を行うことが可能となる。NaraViewは奈良先端科学技術大学院大学で1996年にUNIX環境下でOpenGL/MOTEFを使用して開発された。今回研究の対象としているNaraViewは和歌山大学で1998年にUNIX環境からWindows環境に移植され、グラフィックスライブラリにOpenGLを継続して採用したものである。ここで生じた問題点は描画処理速度が遅いため、視覚化可能なプログラムの規模が限られているということである。本研究の目的はPC上のWindous環境下で使用するツールとして、NaraViewのDirectXによる再実装を進めることである。本稿は次のように構成される。2章ではNaraViewの機能と問題点についての説明、3章では解決策の提案、4章で再実装の構想について述べ、5章ではまとめと今後の課題について述べたい。

2　現在のNaraViewの機能と問題点

NaraViewはProgram Structure View（PSV）とData Dependence View（DDV）から構成されている。

PSVは与えられたソースプログラムの1ステートメントに対して1つのポリゴンキューブを対応させてプログラムフロー、ループ階層、並列度を表現する。

DDVはループ中のデータアクセスとイタレーションの関係を3次元空間上に表現する。ループボディでアクセスされる変数のうちどの変数もしくは配列要素が何回目のイタレーションでどのようなアクセ

ス (リード/ライト) が行われるかをポリゴンキューブで表現する。並列プログラミングで必須情報であるデータ依存情報を視覚的に表すのが DDV の主たる機能である。

データアクセスのループボディの中のイタレーション数を仮に 100 とすると, 同数のイタレーションで3重ループならば 100^3個の 3 次元オブジェクトが生成される。現段階で *NaraView* が視覚化可能なのはサンプルプログラムと同規模の数百ステップ程度のプログラムまでである。研究用の試作品としての完成度は高いが、実際の開発現場で求められている水準 (1 秒あたり数千万ポリゴンの描画処理) には達していない。実用化を実現するためにより高速なグラフィックスライブラリを使用する必要が生じていた。

3 DirectX と OpenGL

2 章であげた問題点の解決法として、我々は DirectX による再実装を提案する。DirectX は Microsoft 社から提供される拡張 API 群であり、元々ゲーム用に開発されたもので、高速な描画処理を特徴としている。

3.1 再実装によって得られる利点

DirectX による再実装を行うことによって、我々は何よりも描画処理の高速化という利点を得ることができる。これにより、視覚化可能なプログラムの規模が拡大する。

3.2 OpenGL との比較

3DAPI ベンチマークテスト PolyEdit[6] を用いて Direct3D と OpenGL の描画速度を計測する。このベンチマークテストでは、データ構造と 3D 描画命令の違いによるパフォーマンスの変化を測定している。使用したカードは Matrox Marvel G-400TV、CPU は Pentium Ⅲ 300MHz である。計測に使用した描画モードは Direct3D が window mode、OpenGL は double buffer mode である。また、DirectX のバージョンは 7.0 である。グラフ中の黒白 2 種類のポールはそれぞれ Direct3D、OpenGL による測定値を表している。

テストは異なるポリゴン数で作られた 3 種類の球体を用いて 128 個の球体モデルを描画し、その描画速度を測定する。T0 から T2 は使用する描画命令の種類で内容は表の通りである。L0 から L2 は球体に使用するテクスチャのトライアングルポリゴン数を表し、順に 80、320、1280 である。

	使用する命令
T0	従来型の描画 API で 1 回の描画命令呼び出しで 1 つの球を描画
T1	インデックス頂点バッファAPI を使用し、複数回の描画命令呼び出しで 1 つの球を描画
T2	インデックス頂点バッファAPI を使用し、1 回の描画命令呼び出しで 1 つの球を描画

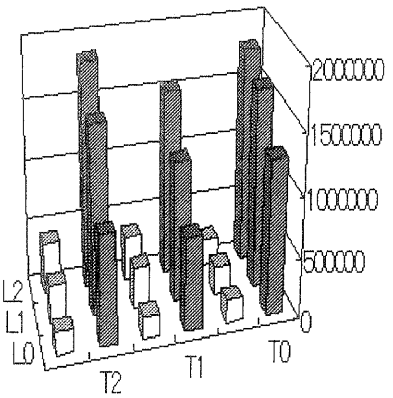

図 1: 1 秒あたりの描画ポリゴン数

項目ごとに計測数値を比較する。まず、フレーム描画速度は Direct3D と OpenGL の差が最大で 7.79 倍、最小で 3.45 倍である。平均は 5.28 倍である。ポリゴン描画速度は DirectX と OpenGL の差が最大で 7.78 倍、最小で 3.44 倍である。平均は 5.26 倍で

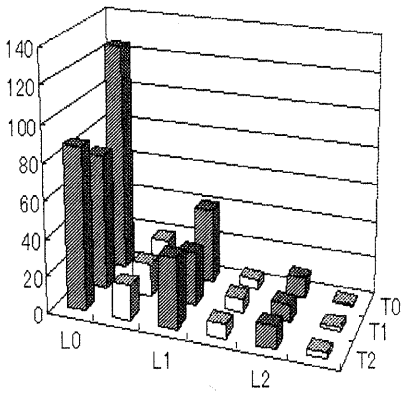

図 2: 1 秒あたりの描画フレーム枚数

ある。

次にポリゴン描画速度を比較する。

最大計測値は Direct3D が 1859964pps、OpenGL が 421849pps であり、このとき使用した描画命令はともに T2 である。Direct3D と OpenGL の計測値の差は 1438115pps であり、これはトライアングルポリゴン 12 個から構成される最小かつ単純な立方体ポリゴン 119843 個を 1 秒間に描画可能な速度である。

Direct3D が最低の描画速度を計測したのは描画命令に T1 を使用したときで値は 756096pps であった。このとき OpenGL の計測値も最低値 183695pps を記録した。計測値の差は 572401pps であり、最小の立方体ポリゴン 47700 個を 1 秒間に描画可能な速度である。

また、Direct3D の最低計測値と OpenGL の最高計測値を単純比較すると、差は 334247pps であり、これは最小の立方体ポリゴンを 27854 個を 1 秒間に描画可能な速度に相当する。

いずれの項目でも OpenGL の測定値が Direct3D の測定値を上回ることはなかった。

以上より、Direct3D を使用することで NaraView の描画機能を大幅に改善できると考えられる。

Direct3D の場合、図形情報を必要に応じてグラフィックチップと CPU で分担して処理しているので、CPU のクロックアップなどの処置で Direct3D の描画処理速度をさらに向上させることが可能と予想している。

3.3 実装上の問題点

OpenGL に比べて DirectX は抽象度が低いためにプログラミングの際に細部にわたって変数の設定を行う必要がある。このため、ともすれば DirectX のプログラムは複雑で難解なものになりがちである。この点に注意してクラスを構築する。

4 NaraView 再実装の構想

互換性を保つため、3D 描画に Direct3D 直接モードを使用する。

MIRAI コンパイラ [5] から出力される中間表現 hierarchical task graph[1] を受け取り、情報を格納し親子関係をそのまま維持してフレーム描画を行うためのクラスを構築する。このとき視覚化された全体の配置は木構造になるため、階層によってポリゴンキューブの間隔を調節し全体のフレームの座標を書き換えるよう、関数を用意しておく。なお、MIRAI コンパイラは日本学術振興会・未来開拓学術研究推進事業・知能情報高度情報処理『分散・並列スーパーコンピューティングのソフトウェアの研究』(代表：京都大学・島崎教授) で開発が進められている。

今回の再実装は DirectX7 で行うが、将来の機能の拡張とバージョンアップを考慮し、2D 描画と 3D 描画が統合された DirectX8 による再実装をハードウェア・ソフトウェアの対応状況を見極めながら計画しておく。

5 まとめ

本稿では並列化支援視覚化システム NaraView を DirectX を用いて再実装する意義を主にベンチマークテストの結果を用いて説明した。

Direct3D と OpenGL の描画速度を比較すること

で、Direct3Dの描画性能が従来のNaraViewに使用されていたOpenGLよりもはるかに高速であることを示すことができた。これによりDirectXによる再実装とその結果もたらされる描画処理の高速化がNaraViewの実用化を実現させるために有効であることが示された。

謝辞

DirectXとOpenGLの比較を行うベンチマークテストプログラムを提供してくださった永谷真澄氏に感謝いたします。

参考文献

[1] M.Girkar and C.D.Polychronopoulos. The hierarchical task graph as a universal intermediate representation. *International Journal of Parallel Programing*,22(5):519-551,1994.

[2] M.Sasakura, K.Joe, Y.Kunieda, and K.Araki. NaraView:AN interactive 3D visualization system for parallelization of programs. *International Journal of Parallel Programing*, 27(2):111-129,1999.

[3] 羽田　昌代, 笹倉　万里子, 城　和貴.

NaraViewを利用した実アプリケーションの並列化事例

情報処理学会　*HPC*研究会,HPC-81-8, pp.39-44,2000

[4] M.Sugita, M.Ohto, M.Sasakura, Y.Kunieda, K.Joe. Intuitive Data Partitioning by a Simple Physical Model and Its Visualization *International Conference on Parallel and Distributed Processing Techniques and Applications*, pp.729-735,2000.

[5] Y.Kunieda, K.Joe, A.Fukuda, T.Uehara, S.Saito, T.Saito, M.Sasakura, T.Nakanisi. *Design and Implementation of an Automatic Parallelizing and Distributing Compiler with Visualization Tools and the Runtime Environment*, Proceedings of the International Conference on Parallel and Distributed Proceeding Techniques and Applications, Vol. II,pp.713-719,June 2000.

[6] Masumi Nagaya. *3DAPI* ベンチマークテスト *PolyTune* , http://www.win.ne.jp/ m-nagaya/index.shtml