MEMOIRS OF SHONAN
INSTITIUTE OF TECHNOLOGY
Vol. 48, No. 1, 2013

３D 立体映像システムにおけるオブジェクトの実時間生成処理

坂下善彦[†]　渡邉優太[‡]

Real time generation processing of the object in 3-dimensional stereoscopic system

Yoshihiko SAKASHITA[†]　Yuta WATANABE[‡]

Abstract:
There is the feature of a 3-dimensional scenography display system in the appearance of a subject (object) not only looking three-dimensional, but a watcher entering inside the subject and being able to see.
We are advancing construction of the design task space of three-dimensional type structure LSI aiming at the directivity using the mobility in three-dimensional space, and the free nature of a viewpoint position.
Although many systems which indicate the object beforehand defined with the modeling tool by visualization exist by a traditional technique, the mechanism which newly defines an object and it adds into the existing object group within continuous time while making it display is underdeveloped.
In this research, the new technique of having used the processor GPU suitable for distributed parallel processing as this solution is proposed.
KEY WORDS : stereoscopic system, real time generation, graphics,data tranformation

要旨:
３次元立体映像表示システムの特徴は，対象物（オブジェクト）の外観が立体的に見えるのみならず，その対象物の内側に観測者が入り込んで観ることができることにある．
我々は３次元空間内での移動性と視点位置の自由性を利用する方向性をめざし，３次元型構造LSIの設計作業空間の構築を進めている．
これまでの手法では，モデリングツールにより予め定義されたオブジェクトを可視化表示させるシステムは多く存在するが，連続時間内で，表示させながら新たにオブジェクトを定義して既存のオブジェクト群の中に加える機構は未発達である．本研究ではこの解決策として，分散並列処理に適した処理系GPUを利用した新たな手法を提案する．

キーワード : 立体映像，実時間処理，グラフィックス，データ変換

1. はじめに

３次元立体映像表示システムの研究を進めている[1]．このシステムの特徴は，対象物（オブジェクト）の外観が立体的に見えるのみならず，その対象物の内側に観測者が入り込んで観ることができることにある．

従来型の３次元表示システムでは，部品関係図の類を介して対象物の構成要素の接続あるいは従属関係を俯瞰図のようにして見ていた．この場合は，特定の構成要素間に注目して，相対的な位置関係を見ている．製造過程における組立加工作業では有効な情報となる．

しかし，構成関係が確立した状態，即ち組立加工作業後の状態におけるその対象物を外側からのみならず，内部の様子を見とおすことはできない．

この故に立体表示空間の内部にこれらの対象物を表示させることにより，対象物の内部の様子も観ることが可能となり，組立加工段階以前の設計段階において，対象物の最終状態を見て特に要素間の状態を確認しながら，設計作業を行うことが可能となり，有用な CAE 状況を提供することが可能となる．

この分野におけるこれまでの研究は，３次元立体映像表示システムの構築を主として行ってきた．立

[†] 湘南工科大学　工学部　情報工学科　教授
[‡] 湘南工科大学大学院　工学研究科　電気情報工学専攻

体表示ができることにより，展示館や博物館，あるいはエンターテイメントの領域での活用がなされてきている．この研究の多くの方向は，よりリヤルな精細できれいな対象物の迫真感ある映像表示を追求する研究が盛んであるが，我々は３次元空間内での移動性と視点位置の自由性を利用する方向性をめざし，この研究環境を基盤として，上記の設計作業空間を構築することの段階へと進めている[2][3].

即ち，製造領域における適用を一義とし，生産現場への利用をめざし CAE(Computer Aided Engineering)を適用の対象としている．具体的には，最近の LSI 設計が立体構造化されていく状況にあることを踏まえて，３次元立体型 LSI の設計に適用できる仕組みとシステム環境を構築する．

2. 3次元表示システム

現在の多くの立体映像表示システムは, 両眼視差の特徴を利用した左右画像分離方式により実現している．我々も，㈱ソリッドレイ研究所のOmegaSpaceを使用して複数スクリーンに表示させる方式で表示環境を構築している．

複数スクリーンに映像を同期させて表示させる必要があり，現在は優れたグラフィックスのハードウェアボードを利用することができ，複数のグラフィックスボードの間の同期処理も提供されているインタフェースを利用することにより，比較的容易に実現することができる．

むしろ, 表示されたオブジェクトを観測者が操作する操作インタフェースに関する課題の方が多いと言える．基本的に，グラフィックスシステムにおいては，オブジェクト定義の基盤ともなるモデリング段階と，この定義されたオブジェクトを可視化させるレンダリング段階に大別される．

しかし，多くの場合において，この両者は連携してない．即ち，モデリングツールにより予め定義されたオブジェクトを可視化表示させるシステムは多く存在するが，連続時間内で，表示させながら新たにオブジェクトを定義して既存のオブジェクト群の中に加える機構は未発達である．

多くの場合，前段となるモデリングのシステム（あるいはツール）と後段の可視化システムは機能的に分離している，あるいは極めて簡便な機能を備えているのみで，基本的に別システムとなっている．先ずは，この点に焦点を当てて連続的処理の実現を目指す．

3. 実時間処理

本論で述べる実時間処理は，グラフィックスの技術領域で多く扱われるデータ処理の高速化を対象としたものではなく，前記のモデリングにより定義されたデータを後段の可視化処理に渡して表示させるこの一巡の処理を高速化させることを対象としたものである．

コンピュータグラフィックスの長い発展の過程では、多種の業務を対象にしたさまざまな機能や形態のシステムやツール類が開発され製造されてきた。そこにおける課題の一つがユーザインタフェースに関わるものであり、他がツール間にやり取りされるさまざまな形式のデータ形式である。前者は、UIMS(User Interface Management System)の概念が考案されて広くこの考え方が取り入れられ、併せて Window System の発展に伴い、その課題の存在は薄れてきている、他方、データ形式の多種性は今日まで引き続きあり、業界における手間のかかる問題として依然として存在している．

コンピュータグラフィックスの処理言語の種類は多くはないが、定義する図形を蓄積するデータベースあるいはファイル形式は、前述のように多種多彩である．このような要素も含めて本論では、データ形式と総称して扱う．

設計および製造の現場においては単一のツールで対応できることはきわめて少なく、複数のツールを扱うことになる．本論では、このツール間のデータの受け渡しに注目する．

例えば、図形オブジェクトを定義するモデリング過程と定義されたオブジェクトを可視化するレンダリング過程と大別される．即ち、予め使用するオブジェクトの基本は定義しておき、後段において編集操作を加えて位置や相互の結合関係などの相対的な位置関係の定義、あるいはオブジェクトの材質や色などに関わる諸属性の定義などが行われる．

しかしながら、作業の途中段階で新たなオブジェクトを定義して追加していく場合は、前述の 2 つの過程にまたがる作業となるためにその実現は容易ではない．

従来方式では，この２つのシステム間で人的な操作やファイル渡しの形態でデータを渡している．この方式では一連の処理操作を短時間内で実施することは不可能である．この故に、連続する時間関係の中で処理ができることが要求される．前述のように単にデータを渡すだけではなく、データ形式の違いを吸収するためのデータ形式の変換が必須となる．

本研究では，デーや形式の変換処理は外して計算

機連携の技術を用いてこの２つのシステムの間のデータ渡しを高速に行うことを実現することが目的としている．

方式１：ファイル渡し

Unix における pipe 機能を使用して、データの授受を行うものである．Unix 環境における構築手段としては容易な手法であるが、基本的にはファイルの授受方式なのでよい性能は期待できない．更に、対象としている画像処理は一定の間隔でデータの集まりを転送処理する機構が主となるため、ファイル共有化がなされている NAS(Network Attached Storage)や各種の記憶システムが統合された形態で且つブロック単位でアクセスできる SAN(Storage Area Network)などの分散型かつネットワーク化されたファイルシステムが存在するが、大規模なファイルデータを扱う形態ではなく、またデータベースシステムのように常時データのアクセスを行う形態ではないので、我々が想定する分散環境下での利用にはそぐわない、と考える．

方式２：メッセージ交換

分散システムの発展によりさまざまなコンピュータ環境での実装構築が容易になってきた方式の一つである．特にネットワークに接続された異機種によるシステムの構成には適している．処理がアプリケーションのレベルで行われ、データはメッセージ化されネットワーク上を転送される．

手法としては、分散メモリの並列計算におけるメッセージ通信のためのライブラリの規格 OpenMPI(Message Passing Interface)が存在し、最近のグリッドコンピュータシステムやスーパーコンピュータシステムでの利用が盛んである．一群のデータを区間分割して並列的に転送させる機構を組み入れる場合に有効な手段になりえる．

我々のこれまでの研究[4]においても、有用性が高いことを確認している．疎結合されたシステム間で並列処理を実施する場合においても，ノードあるいはプロセッサの間での処理の同期を制御することも容易であり有効である．

方式３：メモリ渡し

分散システムを構成するシステム間で共有するメモリを設定し、システムレベルでの処理によりデータをこのメモリを経由して授受する方式で，ユーザレベル，システムレベル，OS レベル，等で提供されている共有メモリ方式が主な対象となる．基本的に１回の送信あるいは受信につき、メモリへの書き込みとメモリからの読み出しの２回の処理を要することになるために処理手順としての効率は悪いが、OS レベルでの処理が可能なことから時間的な処理性能が期待できる[5]．

我々のこれまでの研究[6][7][8]では、疎結合された分散型システムにおいてデータを共有する場合においても有効な手法であることを確認している．

最近のマルチコア CPU や次に述べる GPU のアーキテクチャにおいても，この共有メモリの機構は多く採用され，プロセッサ間あるいはスレッド間でのデータの共有により処理効率を上げる有効な手法となっている．

方式４：GPU の活用

以上述べたデータが経由する経路に注目した伝統的な手法とは別に，本研究における特徴となる方式である．

本研究における焦点は，高速なデータの移送と変換である．即ち，通信路とデータ蓄積と処理を効率よく実施することである．グラフィックスの処理エンジンとして多く採用されている GPU を利用することにより，データ蓄積と処理の部分を対応できると考えている．この詳細は５章にて述べる．

4. データ形式

本研究は，既に３次元立体型 LSI の設計がモデリングツール Metasequoia にて行われていること，および可視化のシステムは OmegaSpace(ソリッドレイ研究所㈱)を用いているので，本研究ではこれらのツールを対象として想定している．

4.1 Metasequoia のデータ形式

(1) 拡張子

MQO:オブジェクト
MQM:材質
MQP:画像ペイントのパレット
MQB:画像ペイントのブラシ

(2) MQO:オブジェクトファイル

Header, Scene, Material, Object, Blob, などの以下に挙げる親チャンクで構成される．

Header
TrialNoise
IncludeXml;プラグインの情報の保存用
Scene;視点情報など
BackImage;下絵情報など
Material;材質情報
Object;
Vertex;頂点群
Vertexattr;頂点毎の属性，uid, weit, color
Bvertex;vector, weit, color
Face;頂点インデックス，材質インデックス，UV

値，頂点カラー，Catmull-Clark 曲面用エッジの折れ目
Blob;メタボール用
(3) MQM 材質ファイル
MQO と同じで Material のみ持つ
(4) MQP パレットファイル
マッピング画像のペイントに用いられるパレットを定義
(5) MQB ブラシファイル
マッピング画像のペイントに用いられるブラシを定義

4.2 既存ツール類の組合せの調査
この類の課題は，所謂 CAD システムが誕生して以来の未だに存在するものである．ユーザインタフェースに係る問題は UIMS などの研究や，最近のコンピュータ環境はほとんどウィンドウシステムが搭載されていることで概ねは解決しているが，グラフィックスの関連言語が幾つか登場してきているがデータの授受に関する問題は依然として継続している．
仮想オブジェクトモデル DOM のような概念が存在するがこれはある程度統一された実行環境の存在が必要となり大きな制約事項となっている．
現在のグラフィックスに係る領域では，統合化されたデータ環境はない．この故に，ツール間の連携は個別対応となっている．本研究においてもこの制約は免れないが，比較的連携関係の良い組み合わせを求めることとなり，研究対象領域ではない．

Table.1 モデリング・ツール metasequoia

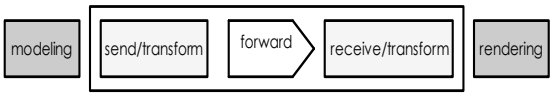
API 仕様(Out)	データ形式
LightWaveObject	*.lwo
3D Studio	*.3ds
AutoCAD	*.dxf
POV-RAY	*.pov

Table.2 可視化・ツール OmegaSpace

API 仕様(In)	データ形式
LightWaveObject	*.lwo
3D Studio	*.3ds
VRML2.0	*.wrl
MicrosoftDirectX	*.x

5. 提案方式の検討

5.1 基本構成
(1) 基本構成の概念
３章にて述べたように，基本的に２つの役割を持つモジュールあるいはツールが存在する．この両者の間の連携を高速に処理する仕組みを検討する．その基本構成は図１に示すようになる．基本的に何らかのデータ変換が付きまとうことを考慮する，それを転送する前に行うか転送後に行うかはシステムの特性に依存する．

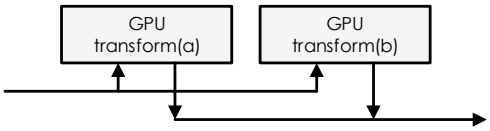


Pic.1 基本構成

(2) 同時処理・実時間処理用にダブルバッファ方式を採用
データはグループを形成してバッチ的に転送するのではなく，常時更新されたデータや新たに加わったデータを可能な限り早い内に反映させることが実時間処理に求められている要請であるので，データが巡回的に転送される形態を前提とする．
この故に，図２に示すように，グラフィックスの領域で多く採用されるダブルバッファあるはトグルバッファ方式を採用する．

Pic.2 ダブルバッファ方式

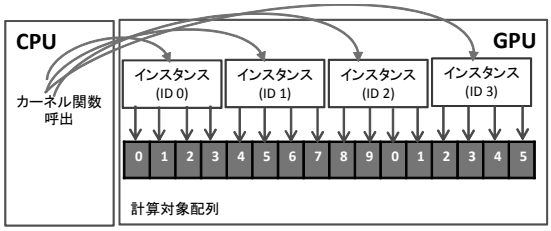
(3) 高速変換処理に GPU を使用する
３章では，いくつかのデータ転送手法を示した．それらは分散並列処理の研究から考案された方式である．将来的には分散並列処理システムを踏まえた構築手法を目指すが，本報告では，実時間処理の実現に重点を当てた対応の検討を示す．
この転送と変換の処理を同一あるいは隣接する場所で実施させることで，両方の処理時間を可能な限り小さくすることを狙うために，変換処理を行う GPU モジュールのフレームの中で両方を実施させる．その基本構成を図３に示す．



Pic.3 GPU を用いる提案方式

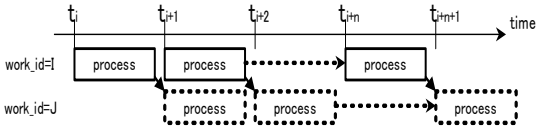
5.2　提案方式

前節で示した GPU を用いる提案方式を，GPU の特性を挙げながら紹介する．基本は，図 4 に示すように，CPU 側は並列処理単位に処理を分割して GPU 側へ処理と関連するデータを GPU 側へ送る．GPU 側では，その処理単位毎にハードウェア資源（基本的にプロセッサとメモリ）に割り当てられて，処理が実行される．この際の実行主体はスレッドと呼ばれる形態のプログラムである．従って，扱うデータの間の独立性が高いほど分割における制約が小さくなる．この故に，グラフィックスにおける処理の多くが配列の計算や行列の計算であり，その要素数は極めて多量であるがデータ間の独立性が高いために，この手法が発展した．



Pic.4. CPU と GPU

今日の多くのグラフィックスの処理は，その多くがレンダリング処理に費やされるためにこの GPU を使用するシステムが多い[9]．もともとこのレンダリング処理が GPU を利用して行われているので, GPU 資源の一部をデータ変換の処理に充てることとした．一般的に GPU においては, 指定する作業量と作業数の総和に従って並列度が決まる[10]. 本研究においては, 図 5 に示すように複数種類の処理を GPU 内部において一様に同時に実施する機構をソフトウェア的手法により可能とすることを目的としている．

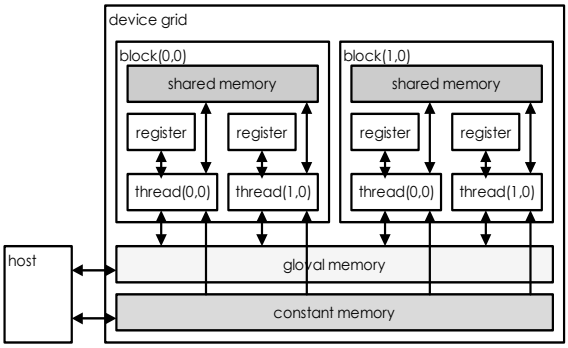


Pic.5. 複数種類の処理を同時順次実行

即ち，図 3 に示したように，同一のモジュールの中で複数（両方）の処理が可能であれば，CPU と GPU の間でのデータの出し入れの操作を行うことにより，この場所でのデータ交換の場として扱うことで，新たに処理のステージを設けることなく実現することが期待できる．
この故に，本報告では，レンダリング処理には触れず，データ転送と変換に焦点を当てる．

5.3　GPU の概要

CUDA が導入したメモリアーキテクチャモデルでは，ホストとデバイスが個別のメモリ空間を持つ．デバイス上でカーネルを実行するために，プログラマがメモリにデバイスを割当て，割り当てたデバイスメモリにホストメモリから適切なデータを転送する．そして，不要になったデバイスメモリを開放する．図 6 に示すように，プログラマがデバイスの様々なメモリタイプの割当て，移動，使用する．



Pic.6. GPU デバイスのメモリモデル

5.4　変換処理の分割と集約

処理性能を上げるために，一般にアプリケーションはデータの並列処理機能を利用する．グローバルメモリから大量データを短時間に処理するために，共有メモリを用いて，スレッドブロックの一群のスレッドがアクセスしなければならないデータの総量を減らすためにタイル分割を行う．これは，レンダリング処理に大きく関係している要素である．

ここでは，データをグローバルメモリから共有メモリとレジスタに効率的に移行するメモリコアレッシング技法に注目し，主にメモリの特性，具体的には帯域幅に焦点を当てる．
グローバルメモリは DRAM で構成されているので，連続的にアクセスすることにより高速にデータを操作することになる．即ち，メモリの帯域幅に可能な

限り近づくようにメモリアクセスを調整することが望まれる．従って，全てのスレッドに対する同じ命令がグローバルメモリの連続した場所にアクセスすることである．

5.5　GPU 内メモリ間でデータ転送

GPU にはいくつかの種類のメモリが備えられているが，CPU 側から操作できるメモリは図５に示すグローバルメモリとコンスタントメモリである．

デバイス側でできるのは，以下である．

- スレッドごとのレジスタの読み書き
- スレッドごとのローカルメモリの読み書き
- ブロックごとの共有メモリの読み書き
- グリッドごとのグローバルメモリの読み書き
- グリッドごとのコンスタントメモリの読込み

ホスト側でできるのは，以下である．

- グリッドごとのグローバルメモリとコンスタントメモリ間のデータ転送

5.6　提案方式の基本処理性能の予備計測

提案する方式のベースとなる予備実験を示す．この手法は，複数のモジュールあるいはツール間をデータが移動すること，そして何らかの変換処理が必要なことを踏まえて，この移動と変換の処理を同一あるいは隣接する場所で実施させることを狙っている．以下に使用したコンピュータシステムの仕様を示す．

CPU: Core i5-2400@3.10GHz × 4
GPU: GeForce GTX 660
Memroy: 4GB
OS: Ubuntu 12.0.4 LTS
Blender 2.68a
CycleRender CUDA5.5

(1) GPU の処理性能概要

予備計測として現有の GPU の処理性能を見極めることを実施した．OmegaSpace の利用インタフェースや内部構造が公開されてないために，処理速度の観測ができないために，ツール”Blender[11]”によるレンダリング処理の時間を観察した．GPU を利用する場合は，利用しない場合に比べて 4 倍近い処理性能が得られている．但し，GPU にて純粋に処理している時間の観察はできてなく，データの分配と集約時間を含んでいる．

Example; PathTracingTile
[sample=200, x=480, y=270(４分割)]
GPU=1:03:30[min]
CPU=3:44:57[min]

(2) GPU 単独の処理性能

GPU における処理時間を図５に示した形態にて，作業量と作業数の関係から観た．図７に示す処理を GPU に１～１０万回実行させ，その処理時間の平均値は，3.16～ 3.29μSec/処理となった．

```
#PyOpenCL(Program Object)
prg_theta = pyopencl.Program(ctx, """
    __kernel void theta(__global float *t, float n)
    {
        int gid = get_global_id(0);
        t[gid] = 2.0 * M_PI * gid / n;
    }
    """).build()
```

Pic.7. GPU 処理プログラム

(3) CPU と GPU 間データ転送性能

データの転送性能を観るために，CPU から GPU 側のメモリへ float 型のデータ 1024 個を転送して書込み，GPU 側で処理を行った後に，同数のデータを GPU 側のメモリから CPU へ読み出して取り込みまでの時間を計測した．その平均転送時間は，50.64～52.48μSec/送受信転送 となった．

計測の(2)と(3)の結果を表３にまとめる．

Table.3 計測結果（１回当りの平均値，単位=μSec）

計測回数	全体処理	転送＋GPU 処理	GPU 処理
10x10^3	81.70	50.64	3.77
100x10^3	82.18	51.15	3.16
1000x10^3	83.93	52.48	3.29

6. 提案方式と予備計測結果の評価

両眼視差を利用した立体映像表示システムにおいては一般に 60Hz×2=120Hz/frame の速度，即ち，8.33mSec/frame でフレームデータを表示する．換言すると，この時間内に１フレーム分の処理を完了させなければならない．その多くの部分は，レンダリング処理の時間と予想されるので，データ変換に許される時間は極めて短い．

予備計測の結果から，データ変換処理は 55μSec 以内で処理でき，関連する CPU の側での処理時間は更に 30～50μSec 費やされるので，総計では 80～100μSec と予想できる．CPU と GPU 間のデータ転送時間は，ツールのデータ量から 2～4KB 程度とみられるので，今回の予備計測の結果からも，合計 100μSec 以内に処理できる見通しを得る事ができた．これは，上記の 8.33mSec に対して 1.2%以内の値となり，処理時間の観点からは十分な値と考える．

7. おわりに

我々は３次元立体映像表示システムの移動性と視点位置の自由性の特徴生かして，３次元型構造 LSI の設計作業空間の構築を進めるために，グラフィックス処理のモデリングツールにより予め定義されたオブジェクトを可視化表示させるこれまでの手法ではなく，連続時間内で，表示させながら新たにオブジェクトを定義して既存のオブジェクト群の中に加える機構の実現を目指してきた．

本報告では，３次元立体映像表示システムの概況を紹介し，システムを構成する上で要となるアーキテクチャに焦点を当てて，その特徴について述べた．

そして，本報告の主題となる連続時間内での映像オブジェクトの生成から表示に至る一連の処理過程を可能とするための，分散並列処理に適した処理系 GPU を利用した新たな手法を提案した．

この実現可能性を見極めるために，CPU と GPU の連携処理の実行時間を予備的に計測した．この結果，全体の制約条件の数%以内で，変換処理が可能である見通しを得ることができた．

これにより，モデリングの段階から新規にオブジェクトを生成してから，モデリングのデータをレンダリング処理へ変換処理を行いながら渡すことが制約時間内で実現できる可能性を得ることができた．

今後の課題は，変換用のデータに関するプロファイル情報の定義，データ変換のアルゴリズム設計，そして実現手法と実装である．

参考文献

1) 佐藤，二宮，坂下：３D 立体映像オブジェクト操作用インタフェースの開発―リモコンによる傾きと高さ位置の検出―，情報処理学会第 74 回全国大会, 1ZD-1,2012.3
2) Sakashita, Watanabe: Optimal control between the object position and view point in the 2 screen's L-type stereoscopic display system, IEEE/GCCE,2013.10
3) 渡邉，佐藤，坂下：2 面 L 型ディスプレイによる立体表示システムの試作，情報処理学会マルチメディア、分散、協調とモバイル(DICOMO2013)シンポジウム,1F-2,2013.6
4) 工藤，坂下：Beowulf 型クラスタシステムにおけるメッセージの振るまい観測と分析，情報科学技術フォーラム FIT2009,M-025,2009.9
5) Hyun-Wook Jin, Sayantan Sur, Lei Chai and Dhabaleswar K. Panda: LiMIC: Support for High-Performance MPI Intra-Node Communication on Linux Cluster, Technical Report of the Ohio State University, OSU-CISRC-10/04-TR58
6) 工藤，坂下：分散共有メモリ間通信方式のクラスタ型並列処理機構，情報処理学会第 72 回全国大会, 2L-5,2010.3
7) 富田，二宮，吉田，坂下：分散共有メモリ方式による情報場の構造化と実装，情報処理学会第 72 回全国大会, 6ZB-7,2010.3
8) 富田，坂下，大谷：分散共有メモリ JavaSpaces を階層構造化した情報場の管理と操作，情報科学技術フォーラム FIT2009,M-050,2009.9
9) Michael Garland and David B. KIRK:Understanding Throughput-Oriented Archtectures, Communication of the ACM, Vol.53, No.11, pp58-66
10) David Koller and Marc Levoy: Protecting 3D Graphics Content, Communication of the ACM, Vol.48, No.6, pp74-80
11) http://blender.jp/ツール"Blender" 3DCG アニメーションを作成するための統合環境アプリケーション