特集　デザイナを中心に置いた，新しいHMIソフトウエア開発手法：グラフィックメータ向けリアルタイム3Dレンダリングエンジンの開発*

Artist-Centric New HMI Software Development Workflow: Development of Real-Time 3D Rendering Engine for Reconfigurable Instrument Clusters

西川良一 Ryoichi NISHIKAWA
中田和行 Kazuyuki NAKATA
梶田健太郎 Kentaro KAJITA
松本壮樹 Soju MATSUMOTO
石川　巧 Takumi ISHIKAWA
渡良井葉麻 Yoma WATARAI

Instrument clusters that display all information on a TFT-LCD screen, also known as reconfigurable instrument clusters, have become the new trend in automotive interiors. DENSO mass-produced the world's first reconfigurable instrument cluster in 2008. To satisfy customer requirements, large quantities of resources were required. Coupled with an iterative process due to requirement changes, development costs became very high. Reducing development costs was vital in order to expand the reconfigurable instrument cluster products-line. Therefore, DENSO developed a new workflow and technology for the reconfigurable instrument cluster. An artist-centric approach was proposed to reduce the development effort by introducing a data converter and real-time 3D rendering engine. Using automatic shader code generation and frame rate improvement techniques contributed to rendering a high quality image on a TFT-LCD screen. The new workflow was evaluated during a mass production development project and resulted in improved software development efficiency.

Key words：Instrument Cluster, Computer Graphics, Model Based Design, Human Machine Interface, REMO

1．はじめに

ゲージを含めたすべての計器をグラフィックにて表示するメータが各自動車メーカにおける旗艦カーの新しいトレンドとなりつつある．一般車では，ここ数年で小型ディスプレイを搭載したメータが主流になっている．デンソーは2008年に世界で始めてフルグラフィックメータを量産化した．それは1280x480の高解像度12.3インチTFTディスプレイを使った製品である．

製品化にあたり，最も大きな問題はハードウェアの制限である．なぜなら，信頼性を確保するためには，大容量のHDDやNAND FLASHを使用できなかったからである．メモリ資源の制限は厳しく，2008年には搭載できるROMサイズは16MB足らずであった．

量産化で先行する，小型ディスプレイ搭載メータの表示は固定画像のコピーによってアニメーションを生成していた．この手法の場合，表示部品をすべてイメージデータとして持つ必要がある．そこで，データサイズを縮小するための技術，例えば，イメージを圧縮する，あるいは，アニメーション中に更新されるイメージに限定して切り取り部品化するなど，工夫する必要がある．しかし，高解像度ディスプレイに対して，これらの技術適用で十分な効果は得られなかった．

具体的には，**Fig. 1**にコマ毎の画像を持ってオープニングムービー（1280x480pixel, 20 fps, 5.2 sec）を制作した場合の概念を示す．単純にすべてのイメージデータを持つと，全体のムービーサイズ合計は256MBにもなってしまう．更新イメージ限定の部品化を適用しても，サイズの合計は178MBである．利用可能な16MBからは，かけ離れた数値である．

思い切ったデータ削減を達成させるために，我々はOpenGLを使用することを決断した．それは，3D（3次元）のCG（コンピュータグラフィックス）を表現するためのクロス言語およびマルチプラットフォームのAPIである．現在では，ほとんどの新しいGPUでデフォルトサポートされているが，2008年当時，組み込み向けとしては搭載が始まったばかりの技術であった．OpenGLの使用によって，頂点情報から3D CGを直接描画することができる（リアルタイムレンダリン

*SAEの了解を得て，SAE2013-01-0425を和訳，一部加筆して転載

グと呼ばれる）．必要なデータサイズはイメージデータを直接持つ場合より格段に小さくできる．実際に，この手法によるオープニングムービーのデータサイズはコマ毎の画像によって制作されたアニメーションと等しい画質にもかかわらず，わずか3MBで実現できた．また，OpenGLは移植性の高さがメリットである．APIは標準化されているため，既存のソフトウエアを移植することにより，他のOpenGL対応のハードウェア上で同じ動きが再現できる．例えば，PC上のシミュレーションから，ターゲットハードウェアへシームレスに移行することができる．もちろん，ハードウェアが変わる場合の移行も簡単である．

開発を通じて明らかになった課題は，開発工数の低減である．顧客の要求を満たすためには多くのソフトウエアエンジニアが必要だった．変更要求が度重なる，いわゆるスパイラル開発により，非常に大きな工数となった．なぜ我々はスパイラル開発をする必要があるのだろうか．「高品質グラフィックのアピールがしたい．」，「車両モデルの毎の差別化がしたい．」という顧客要求がある．しかし，ハードウェア制限のために，顧客からの要求をすべて満たすことは難しい．我々は顧客と共に実際のターゲット上で，描画結果を確認しながら，納得のできる表示を見いだすことが必要である．グラフィックメータの高い柔軟性と車の急速な高機能化により，開発規模の急増加と複雑化は避けられない．顧客とより高いレベルの製品を追求するためには，開発効率を上げ，スパイラル開発に対応できるプロセスを獲得しなければならない．

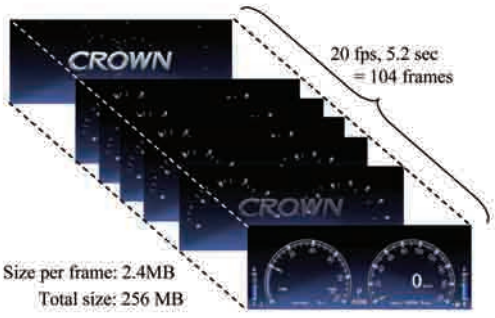

Fig. 1　Implement size of animation by bitblt

2．開発費の分析

開発工数の低減は，グラフィックメータの搭載を拡大するためにも重要である．拡大を実現するために，我々は従来のメータと同じレベルまで開発工数を下げる必要がある．現状では，搭載機能にもよるが，ディスプレイのないメータの開発工数は40人月程度，小型ディスプレイを装備しているメータは100人月程度である．一方で，2008年のグラフィックメータの開発工数は400人月を超えるものであった．我々の最終目標はグラフィックメータの開発工数を100人月以下に下げることである．

Fig. 2に示す，開発工数の内訳を見ると，グラフィックアプリケーションの開発が最大であることが分かる．工数増大の主な理由は，ハンドコードで機能実装していたからである．グラフィックのアプリケーションコードを画像データから自動的に生成することができれば，工数を下げることができる．

経験上，仕様変更によるやり直し工数が多い事が分かっている．2008年には度重なる大規模な仕様変更があり，変更のたびにソフトウエアを1から作り直していた．また，製品が開発されるまで，見栄えを確認することができなかったため，フィードバックするのが遅くなってしまっていた．これには，仮想試作を利用することによって，要求定義と分析の段階においてPCシミュレーション上で見栄えを確認することができれば，製品完成まで待つこと無く，フィードバックすることが可能となり，その結果，大きなやり直しを防ぐことができ，開発工数を下げることが可能となる．

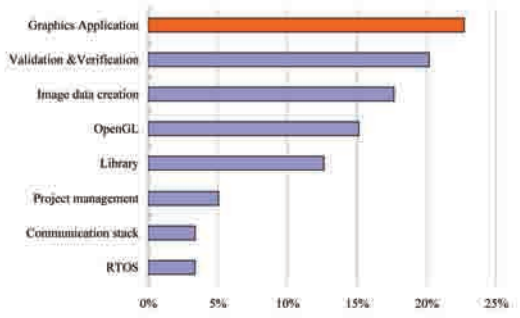

Fig. 2　Breakdown of the development cost

3．HMIツールの課題

開発効率を向上する1つの解決法は，既存のHMI（Human Machine Interface）ツールを使用することである．HMIツールはグラフィカルなHMIの開発を支援する．HMIツールにはCGコンテンツ作成，様々な組込プラットフォーム向けのコード自動生成，シミュレーションおよびドキュメント生成の機能を備えている [1)]．世界には多くのツールが存在するが，どれも業

界標準にまで至っていない．グラフィックさらに，それぞれのツールにはイメージやシーン作成のために独自の方法が存在する．デザイン業界のCGソフトに慣れているデザイナが，それぞれのツールを習得し，使用するのは非効率，困難である．デザインデータ（例えばAUTODESK社 3dsMAXで作成したモデルデータ）をインポートできるツールはある．しかし，ほとんどの場合，シーン構成作業はその後で必要となり，それぞれのツールには行うべき独自の方法が存在する．

その結果，ソフトウエア開発者がデザイナの作ったデザインをツール上で作り直す必要があるため，デザイナから実機までのワークフローは作り替えに苦しむこととなる．設計が変更される度に，デザイナと開発者の両方が再度工程を繰り替えさなければならない．

4．デザイナを中心とした新しいHMIソフトウエア開発手法

メータはドライバの最も重要な情報源であるだけでなく，人と車の間に，ある種のパーソナルな関係性を構築する役割，ドライバと車をつなぐ役割がある．メータのデザインは，明快さ，正確さ，そして美しさを反映したデザインであることが求められる．本論文では，データコンバータおよび3Dのリアルタイムレンダリングエンジンの導入により，開発工数を低減したこと，そして，デザイナ中心のアプローチで開発することを提案する．最終目標は，コンセプトから製造までのソフトウエア開発工程におけるロスを最小化して開発することである．デザイン業界標準のCGソフトで表現されたデザイナのアイデアが，実際のターゲット上で正確に再現されなければならない．

4.1　コンセプト

従来のワークフローを**Fig. 3**に示す．CGデザイナはCGのシーンを作り，ソフトウエア開発者はデータを作り直し，グラフィックアプリケーションを実装する．その後，実ターゲット上のレンダリング結果を確認しデザインにフィードバックすることができる．そのため，結果確認により問題を見つけた場合，CGデザイナおよびソフトウエア開発者の両者が変更しなければならない．このワークフローは，たとえその問題がCGデザイナの仕事であっても，ソフトウエア開発者も修正しなければならない．したがって，効率が悪くなる．

デザイナ中心の新しいワークフローを**Fig. 4**に表す．このワークフローにおいて，データコンバータを用いることにより，CGデザイナによって作成されたCGデータを実機実装可能なデータへ変換できる．変換されたデータはレンダリングエンジンで実行することができ，実ターゲット上でのレンダリング結果をチェックすることもできる．問題がある場合は，CGデザイナだけでCGシーンデータを修正し，改訂できる．

さらにプレビュアの使用により，PCでも結果を確認することができる．我々は仮想試作としてこの仕組みを使用し，開発の初期段階のうちにCGデザイナにフィードバックすることができる．CGデータをインポートできる多くのツールがあるが，それらの多くはアニメーションのシーンデータをサポートしていない．新しいワークフローの特徴は，CGデザイナの成果物を直接利用することができるということである．データコンバータとリアルタイム3Dレンダリングエンジンの詳細は後述する．

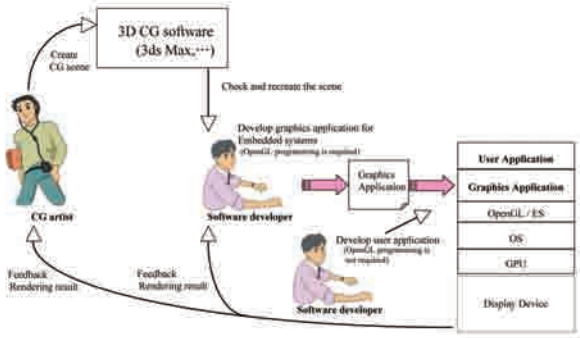

Fig. 3　Conventional workflow

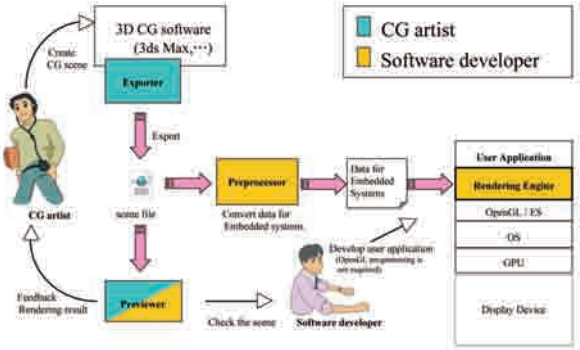

Fig. 4　Artist-centric new workflow

4.2　開発手順の詳細

シーン作成からレンダリングまでの作業過程，構成要素および構成要素間のデータフローを**Fig. 5**に示す．新しいワークフローの詳細は，以下の通りである．

最初に，デザイナは3D CGソフトで3Dモデルとシーンを作成する．（モデリングステージ）次に，3Dモデ

ルとシーンからシーンデータをインポートしてからレンダリングエンジンで使用するためにデータ変換する．変換されたデータは，レンダリングエンジン用のシーンデータおよびシェーダプログラムを含んでいる．（変換ステージ）

最後に，レンダリングエンジン上でデータを実行する．レンダリングエンジンは，変換されたデザイナのデータと開発者のアプリケーションコードからの動的な入力データに基づいて適切なOpenGL ESコマンドを呼び出す．（レンダリングステージ）

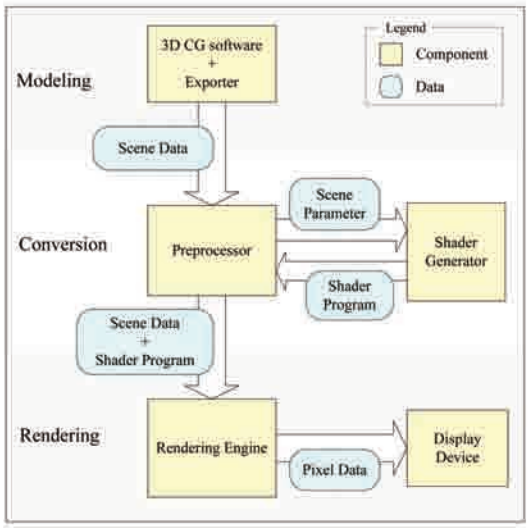

Fig. 5　Overview of the work process, components and dataflow

4.3　データコンバータ

データコンバータは，エクスポータとプリプロセッサから成り，エクスポータは3D CGソフト（AUTODESK社 3dsMAX）のプラグインソフトである．またそれは，環境非依存のシーンデータを出力する．プリプロセッサは，実ターゲット環境用にシーンデータを変換する．

モデリング段階の出力であるシーンデータは，レンダリングステージで必要とされるシェーダプログラムを含まない．したがって変換ステージの出力としては，変換されたシーンデータだけでなくシェーダプログラムも必要である．

シェーダプログラムは，変換ステージでシェーダジェネレータモジュールから生成される（**Fig. 5**を参照）．このモジュールのための入力はシーンパラメータ（主にオブジェクトのマテリアル情報とオブジェクトを輝かす光源の情報）である．このモジュールの出力は，シェーダ言語（GLSL）で書かれたソースコードとそれを実行するための情報である．

4.4　リアルタイム３Ｄレンダリングエンジン

レンダリングエンジンは，変換ステージで生成されたシーンデータを使い，表示デバイスに描画する．あるオブジェクトをレンダリングする時，オブジェクトにリンクされるシェーダプログラムが使用される．レンダリングエンジンは，シェーダジェネレータによって生成された情報と現在のシーン状態を基にシェーダパラメータを計算する．その後，エンジンは，パラメータを設定し，GPUを制御するためにグラフィックAPIを呼び出す．

レンダリングエンジンは速いレンダリング，軽量さ，高い移植性と拡張可能性，のようなニーズを満たさなければならない．

4.5　ソフトウエア開発者の役割

ソフトウエア開発者の役割は，アニメーションを制御することだけである．アニメーションには2種類ある．あらかじめ定義されたアニメーションとプログラマブルアニメーションである．あらかじめ定義されたアニメーションの場合，3D CGソフトでつくることができ，アニメーションのためのレンダリングフレームはユーザアプリケーションから直接指定できる．プログラマブルアニメーションの場合，パラメータは各々のオブジェクトのために設定することができる．例えば，速度計の針の動きを表現するためにCGデザイナが各々のフレームに最小値から最大値まで一定の速度で針を動かす3Dアニメーションを作成する．ソフトウエア開発者は，レンダリングエンジンAPIを呼び出し，現在の値に対応するアニメーションを描いたフレームを指定するアプリケーションコードを書くだけである．0km/hから180km/hにかけての針の動きがフレーム0からフレーム180へのアニメーションとなる場合に100km/hを表現するときは，APIをパラメータ100で呼び出すだけであるため，OpenGLのプログラミングは必要とされない．

4.6　利点

この新しいワークフローでは，デザイナのどんな変更でも，すぐに，そしてシームレスに実ターゲットに適用される．

そのうえ，モデルステージの出力を基にPC上で描画できるプレビュアを用いて仮想試作ができる．特徴は，プレビュアが実際のターゲット環境向けに特殊化

されるため，CGデザイナは簡単に汎用PC上で実組込システムのレンダリングを機能させ，出来栄えを確認することができるのである．

5．技術課題

レンダリングエンジンは，デザイナの意図したデザインにできるだけ近いかたちで組込プラットフォーム上にて，変換データを解釈（実行）することが重要である．

TFTディスプレイへの高品質な描画の達成に貢献する技術的な取り組みとして，シェーダコード自動生成，および，フレームレート向上手法を本論文では取り上げる．

5.1　プログラマブルシェーダーコードの自動生成

高画質の実現手法として，シェーダコード自動生成技術を紹介する．シェーダは見た目に大きな影響を与える．データコンバータは，3DCGソフトでデザイナが調整できるパラメータを元にシェーダコードを自動的に生成，最適化できる．3dsMAX上のシーン設定から直接生成できることがポイントである．シェーダはグラフィックハードウェア上で描画エフェクトを計算するのに主に使われるコンピュータプログラムである．シェーダは，ジオメトリ変換，ピクセルシェーディング機能やカスタムエフェクトを実現するGPUプログラマブルレンダリングパイプラインをプログラムするのに使われる．OpenGL，OpenGL ESの公式シェーディング言語はOpenGL Shading Language（GLSL）である．シェーダジェネレータはブリンシェーディングのパラメータを元にシェーダコード生成と静的最適化を自動的に実行できる．ブリンシェーディングモデルは有名なシェーディングモデルの一つであり，十分リアルであり，処理負荷がそれほど高くないためリアルタイムレンダリングに適しているといわれている．多くの3DCGソフトはこのパラメータをもつ．3dsMAXにおいては標準マテリアルのデフォルト設定はブリンシェーディングである．

シェーディング言語はプログラミング言語として定義されているためハンドコーディングは可能だが，プログラマブルシェーダーは自由度が高いため，所望の表示を達成するには高度な知識，技術が必要である．シェーダ自動生成ができることにより，グラフィクスデータの制作過程を簡素化し，高品質な3DCGシステムの開発コストを削減できる．また，デザイン制作者の教育コストの低減を実現することもできる．

5.1.1　コンセプト

5.1.1.1　シェーダジェネレータモジュール

シェーダジェネレータモジュールは**Fig. 6**に示す手順でシェーダプログラムを生成する．

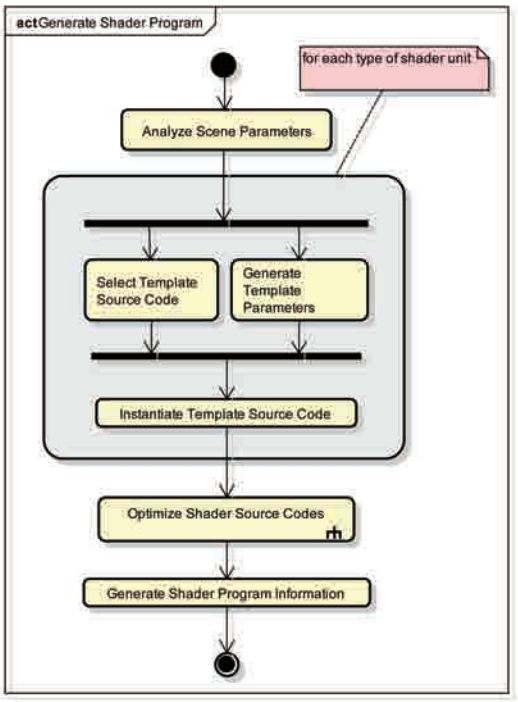

Fig. 6　Procedure of shader generation

シェーダのソースコードは独自形式のテンプレートソースコードを基に生成される．そのテンプレートソースコードをインスタンス化するためにいくつかのテンプレートパラメータが必要になる．それらのパラメータを決定するために，先ず，シーンパラメータを解析し必要な情報を収集する．

次に，それらの情報を基に，テンプレートソースコードを選択し，同時にテンプレートパラメータも生成する．そしてこのテンプレートパラメータを指定して，この選択されたテンプレートソースコードをインスタンス化する．このテンプレートのインスタンス化はシェーダユニットタイプ毎に行われる．シェーダユニットの種類は一般的に頂点シェーダとフラグメント（ピクセル）シェーダがある．

次に，生成されたシェーダのソースコードは最適化モジュールによって最適化される．最適化の詳細は後述する．

最後にこのシェーダのソースコードを実行する際に，レンダリングエンジンが必要とする情報を生成する．この情報には，そのシェーダが必要とする頂点要素やパラメータは何か，そのパラメータをどのように計算するのか，描画されるポリゴンは背景画像とどのように混合されるのかなどの項目が含まれる．

5.1.1.2　シェーダコード最適化

シェーダコード最適化の利点は以下である．

・複雑な計算が必要なシェーダが作成できる
・レンダリングが高速化される
・アプリケーションの初期化時間が短縮される

1つのシェーダプログラムが使用できるGPU資源の量には限界値がある（たとえば，レジスタや命令の数）．最適化によって使用されるGPU資源の量は自動的に削減される．そのため複雑な計算ができるシェーダを作成できる．

レンダリングの高速化は以下の理由で実現される．

・最適化前のプログラムに存在する無駄なコードが削減されるので実行時間が短縮される
・使用されるGPU資源が減少することによってGPU内の実行の並列度が向上する
・CPU側から設定するパラメータが少なくなるのでCPUとGPUの負担が減少する

シェーダコードの不要なコード削減により，ソースコードサイズが小さくなることで，コンパイル，リンク時間が短縮されるため，アプリケーションの初期化時間が短縮できる．

シェーダコードの最適化は**Fig. 7**で示す手順で実行される．

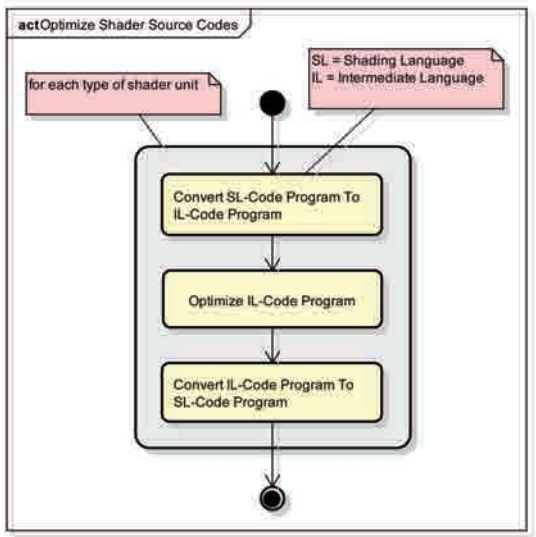

Fig. 7　Procedure of shader code optimization

まず，シェーディング言語のコードを最適化に適した中間言語のコードに変換する．

次に，中間言語のコードを最適化する．実際には不要な処理や変数を削減する．

最後に，中間言語のコードをターゲットのシェーディング言語のコードに再変換する．

この手順によって，ターゲットのシェーディング言語で記述された，最適化されたシェーダコードが生成される．

5.1.2　実験検証

考案手法によって，デザイナの意図したデザインをいかに実機上で再現できているかについて検証した．画質の定量的評価指標として，3dsMAXによる元の描画結果と実機の描画結果のピクセル比較を考えた．実機としてカスタム評価ボードを使用した．ボードはルネサスエレクトロニクス社製のSH-Navi3（SH7776），64MBのROM，256MBのRAM，DVGA（1280x480）12.3インチTFTディスプレイを搭載している．描画結果として，LVDS出力を，インターフェース変換アダプタ（IA-542-B（LVDS to DVI）：ASTRODESIGN Inc.）とコンバータキャプチャBOX（DVI2USB solo: Epiphan Systems Inc.）を用いてキャプチャした．

客観的画質評価メトリクスとして，広く使われているPeak Signal-to-Noise Ratio（PSNR）を用いた．一般的に，PSNRが30dB以上であれば処理された画像は人間の眼で許容できる範囲の画質劣化といえる[2)]．PSNRが40dB以上であれば元画像と処理された画像は人間の眼では差を近くできない．そのため，ここではPSNRが30dB以上であれば画質は十分であるとした．

テスト条件として，1つのティーポットの3Dモデルを様々なパラメータ設定で描画するテストシーンを準備した．各シーンにおいて，3dsMAXの描画結果と実機の描画結果を解像度640x480の32bitビットマップとして取得した．3dsMAXで設定できる一般的なパラメータを全てカバーできるように12種類のテストシーンを選択した．テストシーンはマテリアルパラメータ（ambient，diffuse，specular，shininess，emission，bump，reflection），ライトパラメータ（directional light，spot light，omni light），その他のパラメータ（environment map，skinning，morphing 等）を含む．

5.1.3　検証結果

Table 1はPSNR結果を示す．すべてのシーンにおいて30dBを上回っていることから，十分な再現性を達成できているといえる．主観評価でも元画像と処理された画像の違いの検出は難しいと判断できた．

Table 1　Results of PSNR

Test scene	PSNR[dB]
Ambient	38.68
Diffuse	37.13
Specular	45.64
Shininess	43.10
Emission	34.68
Bump	33.91
Reflection	31.28
Environment map (2D)	38.54
Skinning	33.97
Morphing	34.35
Spot light	38.11
Omni light	39.02

5.2　メッシュ自動連結

高フレームレートを達成する手法として，メッシュ自動連結技術を紹介する．この手法はなめらかな動きの見栄えに寄与する．

5.2.1　コンセプト

グラフィックメータはインジケータ，指針，ダイアル，情報表示のためのパネル，Lane表示，turn by turn（TBT）ナビ表示などを表示する．グラフィックメータの特徴として，描画対象のオブジェクトが大量にあるが，各オブジェクトは単純な形状（長方形）で頂点数が少ないことが多い．

描画コマンド（OpenGLではglDrawElements()）が呼ばれ，描画処理が走るたびに関数呼び出しによるオーバーヘッド時間がかかる．頂点数が多いオブジェクトを描画する際にはオーバーヘッドの占める割合は相対的に小さい．しかし，頂点数の少ないオブジェクトを処理する際にはオーバーヘッド時間が処理時間のほとんどを占める．

描画コマンドの関数コールによるオーバーヘッドは，特に組込環境では描画速度に大きな影響を与える．したがって，頂点数の少ないオブジェクトが大量に存在する場合は，vertex buffer object（VBO）化せず，ひとつの頂点配列に統合し，頂点変換をCPU側で自前計算して，一回の描画コマンドで処理した方が性能改善する場合がある．本手法は，あらかじめ指定されたグループ内の代表メッシュが，代表でないメンバーのメッシュの頂点をまとめて一つの頂点配列として保持・描画する．それゆえに描画コマンドの呼び出し回数が削減できる．

一般的に，各オブジェクトはシーングラフのトラバース順に描画される．シーングラフのデータは構造化されたグラフとして管理される．ノードはリンクによって接続され，ルートノードはシーンデータの始点として定義される．ルートノードからスタートしてノード間のリンクがすべてのノードを通過するまでたどられる．シーングラフのトラバースは深さ優先（兄弟ノードに行く前にすべての子ノードをトラバースする）で実施される．提案手法はこの一般的な処理から逸脱している．

下記の制約を満たすオブジェクト同士はグループ化することができる．

- 同一のマテリアルを参照しているオブジェクト同士のみ同じグループにできる．
- 1つのオブジェクトは一度に1つのグループにのみ所属する．
- 同じグループに所属するオブジェクト同士が重ならない．

5.2.2　データコンバータ

ワークフローにおいて，メッシュ自動連結技術は，データコンバータによるデータ変換時にコンフィグファイル入力オプションを設定して適用できる．コンフィグファイルは各オブジェクトの連結グループの指定を含む．前述した制約のチェックはコンバート時に実施される．

データコンバータは，連結対象のメッシュのどれか一つ（トラバースで最初に出てきたもの）を代表として扱う．代表メッシュはその他のメッシュの頂点データを連結した状態で，かつ非VBOとして保持する．代表以外の連結対象メッシュは実体を持たない代わりに，代表メッシュ内の頂点配列のどこからどこまでがそのメッシュ由来の頂点かを保持する．

5.2.3　レンダリングエンジン

Fig. 8，**Fig. 9**はレンダリングエンジンにおけるメッシュ自動連結技術特有の処理フローを示す．トラバー

ス処理とレンダリング処理がある．

トラバース処理（**Fig. 8**）において，代表メッシュについては，元となる各オブジェクトの情報を元に全ての頂点属性変換をCPU処理し，頂点配列を直接書き換える．ノードの可視性の値については共通シェーダでの処理になる．そのため，連結された各メッシュの連結前のノードの可視性パラメータを考慮しないと代表メッシュの可視性のみが適用されることになる．この問題を避けるため，個々のノードの可視性値を内部的に頂点 α に変換する．また，次に説明するレンダリング処理で使用するための情報としてvisibility flagをノードの可視性をもとにセットする．

レンダリング処理（**Fig. 9**）においては，代表メッシュがもつ連結した頂点のみを描画する．可視性0のオブジェクトについて，頂点 α を0にすることでも対応可能だが，ピクセル負荷がかかってしまうため，オブジェクトを非描画とする処理をしている．具体的には，トラバース処理でセットしたvisibility flagを用いて，インデックス配列上で，描画不要な頂点のインデックスを削除する．元のメッシュの可視性が0でなければ，元のインデックスを，描画用に確保したRAM上の一時的なインデックス配列領域にそのままコピーする．

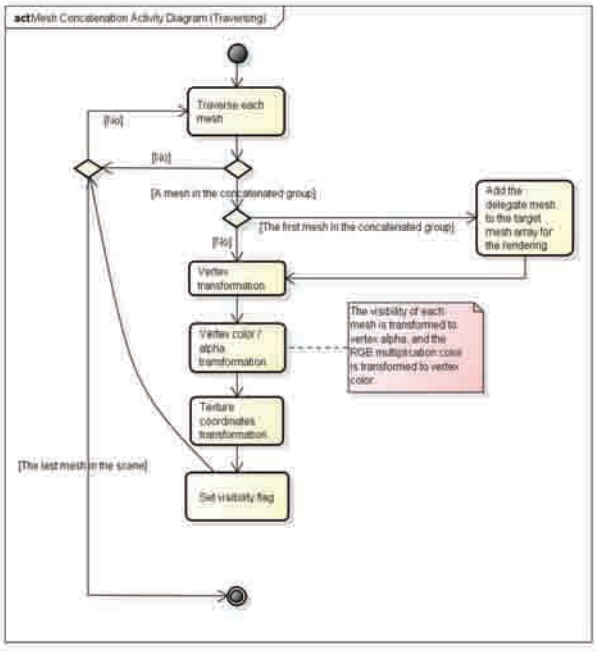

Fig. 8　Mesh Concatenation（Traversing）

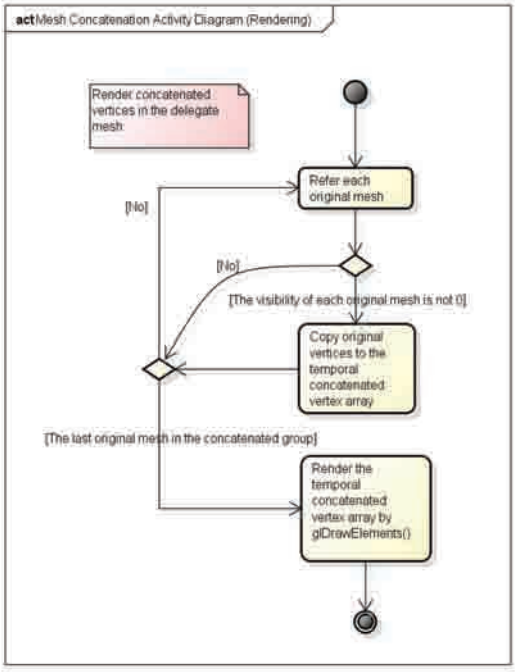

Fig. 9　Mesh Concatenation（Rendering）

5.2.4　シミュレーション

Fig. 10は提案手法により実現される性能改善の理論的モデルを示す．従来，1つのオブジェクトを描画するために描画コマンドを呼ぶたびにオーバーヘッド処理と本処理が頂点に対して実行される．提案手法の適用により，描画コマンドは連結したグループに対して1回のみ呼ばれる．一方，前述したメッシュ連結による特殊処理が実行される必要がある．特殊処理は主に頂点変換なので，処理時間は計算対象の頂点数に依存する．式（1）は理論モデルから導かれる処理時間を表す．oh は1オブジェクトあたりのオーバーヘッド時間，obj はオブジェクト数，v は1オブジェクトあたりの頂点数である．簡単のため，v の値はシーン内のすべてのオブジェクトで一定とする．オーバーヘッド時間は一定値，主処理，特殊処理は v の1次関数と想定した．m が主処理の傾き，s を特殊処理の傾きとする．言い換えると，m，s は各処理の1頂点あたりの処理時間である．提案手法の適用により，2つのダミー頂点がグループ内のすべてのオブジェクトの間に追加される．また，頂点数が偶数のオブジェクトに対し，1つのダミー頂点が追加される．簡単のため，すべてのオブジェクトの頂点数が偶数とすると，1オブジェクトあたり（$v+3$）個を処理する必要がある．

$$\begin{aligned} time_{before} &= (oh+(m \cdot v))obj \\ time_{after} &= oh+s((v+3)\cdot obj)+(m \cdot (v+3))obj \end{aligned} \quad (1)$$

モデル内の全頂点数（v*objで表現される）は提案手法の適用有無にかかわらず一定と想定できる．式（2）は式（1）から導かれる．結果として，提案手法適用前後の処理時間はいずれも1次関数である．

$$\begin{aligned} time_{before} &= oh \cdot obj + m(v \cdot obj) \\ time_{after} &= 3(s+m) \cdot obj + oh + (s+m)(v \cdot obj) \end{aligned} \quad (2)$$

手法適用前の処理時間が手法適用後の処理時間より大ならば，提案手法は効果があるといえる．そのため，条件が式（3）を満たせば手法は効果があるといえる．

$$obj \geq \frac{oh + s(v \cdot obj)}{oh - 3(s+m)} \quad (3)$$

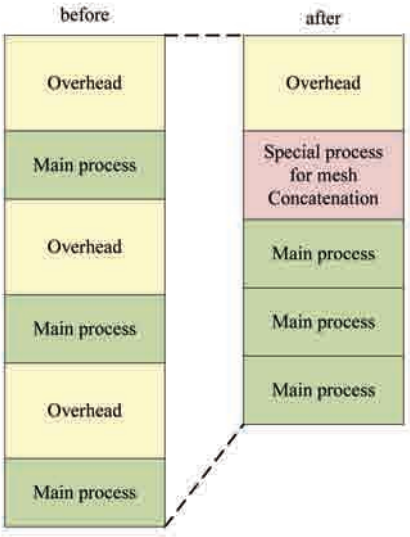

Fig. 10　Theoretical model of the performance improvement realized by the automatic mesh concatenation technique

5.2.5　シミュレーション検証

性能特性が前述したシミュレーションと等しいかどうか確認するため，実機で処理時間を測定した．実機仕様は，シェーダコード自動生成の実験と同じである．

4,000頂点の3Dモデルを準備した．測定条件として，オブジェクト数1，10，100，125，200，250，500，1000とし，オブジェクト構造はオブジェクト数と1オブジェクトあたりの頂点数の積が一定になるように調整した．描画タスクの処理時間を提案手法の適用前後で測定した．CPU負荷とGPU負荷をそれぞれ測定した．長いほうの時間が実際の処理時間である．また，式（2）を用いたシミュレーションのためにoh，m，sを測定した．

5.2.6　検証結果

Fig. 11はシミュレーション結果とCPU負荷の測定結果を示す．シミュレーションに使われたoh，m，sの値はそれぞれ0.16ms，0.00077ms，0.0035msであり，これらは実機で測定された．結果から，理論モデルが実際に測定された性能をよく説明できているといえる．式（3）から，今回使用した実機ではobj > 97 を満たせば提案手法は効果があるといえる．測定結果はオブジェクト数が100を超えたところで提案手法は効果があることを示している．

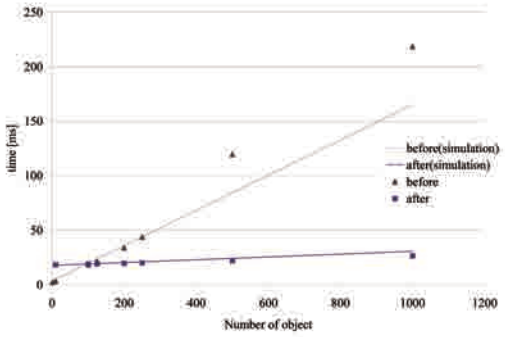

Fig. 11　Result of the measurement of the CPU load and the graph of numerical simulation by using equation (2) with　oh = 0.16, v*obj=4000, m=0.00077, and s = 0.035

理論モデルではGPU負荷を考慮していない．**Fig. 12**はGPU負荷の測定結果を示している．GPU負荷の特性については更なる調査が必要ではあるが，今回の場合は提案手法適用前後でそれぞれ30ms，35msほどである．**Fig. 13**は実際の処理時間（CPU負荷とGPU負荷のうち長いほうの時間）を示す．結果から，オブジェクト数が200を超えると提案手法は効果があるといえる．

提案手法は最適化の条件がそろえば効果があるといえる．特にグラフィックメータにおいては単純な描画オブジェクトがたくさんあるため最適化の条件に当てはまり易い．

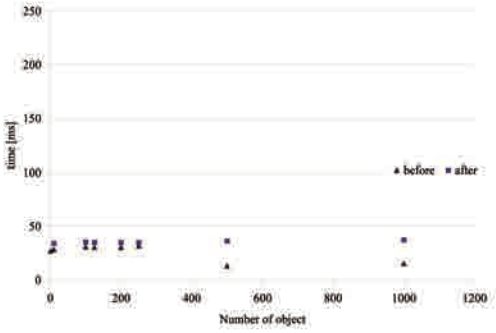

Fig. 12　Result of the measurement of the GPU load

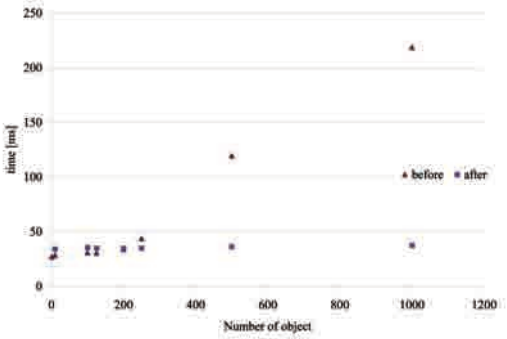

Fig. 13　Result of the processing time (the longer time between CPU and GPU loads)

5.2.7　実機検証例

メッシュ自動連結技術を試作品に適用した効果の例を示す．試作品はルネサスエレクトロニクス社製のSH7769，128MBのROM，256MBのRAM，DVGA（1280x480）12.3インチTFTディスプレイを搭載している．

Fig. 14は試作品の3DCGを示す．コンテンツは4ゲージ（スピード，タコ，燃料計，水温計），テルテール，タブメニュー，運転支援システム表示などである．この3Dモデルは154オブジェクト14,960頂点からなる．いくつかの最適化手法の適用により，17.2msの向上が見込まれたが十分ではない．メッシュ自動連結の適用により，今回の場合は154オブジェクト中の123オブジェクトが8グループに分類できた．**Table 2**は結果の内訳を示す．1オブジェクトあたりオーバーヘッド時間（0.16ms）分の処理時間削減が見込まれるので，提案手法による効果は18.4msとなる．合計で35.6msの削減が見込まれる．

効果確認のため，提案手法適用前後の描画タスクの処理時間を測定した．表示条件は仕様上最も高負荷な条件とした．**Table 3**に測定結果を示す．結果から，GPU負荷に影響を与えずにCPU負荷を下げることができているといえる．なめらかな表示の実現のために高フレームレートが（特に，動きのはやいタコ指針表示に対して）望まれている．人間の視覚特性を考慮すると，少なくとも30fpsの達成が求められる．30fps達成のためには，CPU負荷を33ms以内に抑える必要があり，メッシュ自動連結技術がその達成に寄与した．

Fig. 14　Example of 3D-CG screen by developed product

Table 2　Result of object reduction

Function	Before	After
Telltales	17	1
Speedometer, tachometer and odometer	40	1
Lane guidance	16	1
Turn by turn	16	1
TAB menu	8	1
Fuel and temperature gauge	8	1
Driving assistance system	6	1
Turn list	12	1
Total	123	8

Table 3　Result of the processing time

	Before	After	Improvement
CPU load (ms)	64.0	30.5	33.5
GPU load (ms)	22.5	22.5	0.0

Table 4　Evaluation of the new workflow on mass production project

Vehicle	A	B	Effect
Workflow	Conventional	New	-
Man-month	92.6	95.2	1.02
LOC	6,619,195	113,983,287	17.2
Function	22	50	2.27
Iteration	23	82	3.56

6．量産適用効果

今回考案した新しいワークフローをグラフィックメータの量産プロジェクトにて評価した．**Table 4**は開発コスト，コード行数，搭載機能数およびプロジェクト中のイテレーション（スパイラル？やり直し？）回数を示す．開発コストは開発期間中のアプリケーション開発コストの累積値である．LOCについてはグラフィックアプリケーション部に限定してカウントした．LOCは開発期間中の変更行数の累積値である．

車両Aのアプリケーションコードはハンドコードで実装された．一方，車両Bはすべてのアプリケーション開発に対して新しいワークフローが適用されたため，ほとんどのコードは自動生成された．自動生成コ

ードのほとんどはグラフィック描画のためのパラメータであり，グラフィックコンテンツの規模に従って大きくなる．

結果をみると，新しいワークフローにより，車両Bのプロジェクト規模が車両Aより大きい（LOCは17倍，機能数は2倍，イテレーション回数は3倍）にもかかわらず，車両Bの開発コストを車両Aと同等に抑えることができていることが確認できた．車両Bのプロジェクトにおいて，車両Aと比較して顧客要求の変更が頻繁に発生したが，我々は新しいワークフローにより対応することができた．この新しい開発手法を使うことにより，我々は顧客要望にこたえることができる，あるいは要望以上のことができると考えている．

7．むすび

ツールチェーンを活用したプログラミングレスの開発の達成が理想の世界である．

デンソーはグラフィック開発の自動化を，データコンバータとレンダリングエンジンを用いて達成した．更なる開発費低減に向けて，次のゴールはモデルベース開発ツールとの統合である．

我々のソフトウェアアーキテクチャの基本コンセプトは，よく知られているモデル・ビュー・コントローラ（MVC）である．各パートの独立性を保っており，実際には2つの異なるマイコンに配置していた（**Fig. 15**）．一方のマイコンは車両制御用でありモデルとコントローラを担う．このマイコン上では既存の品質保証済のメータのソフトウエア資産が使われている．他方のマイコンは描画専用であり，ビューを担う．モデルベース開発ツールの適用はコントローラに対してのみ進んでいた．時代のニーズとして，機能や表示の追加，各車種の差別化が求められている．更なる効率化のためには，コントローラとビューを1つのマイコン上に統合し，ツールチェーンを用いて開発プロセスをカバーする必要がある．

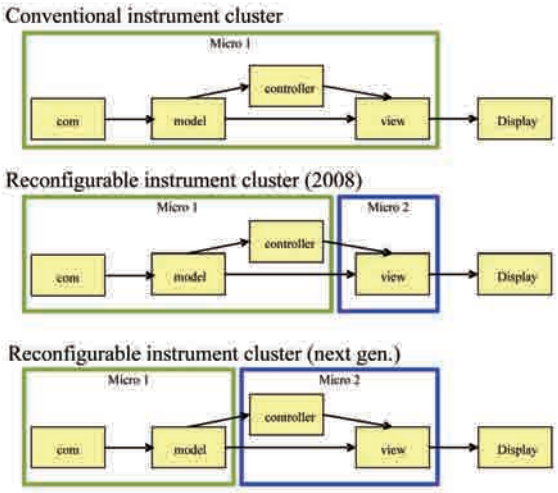

Fig. 15　System block diagram（the past, the present and the future）

＜参考文献＞

1）Y, Lefebvre, “An Embedded Platform-Agnostic Solution to Deploy Graphical Applications” SAE Technical Paper 2011-01-2551, 2011, doi : 10. 4271/2011-01-2551

2）Chen, T.S., C.C. Chang and M.S. Hwang, “A Virtual Image Cryptosystem Based upon Vector Quantization” IEEE Transactions on Image Processing, 7: 1485-1488, 1998, doi : 10. 1109/83. 718488

＜著　者＞

西川　良一
（にしかわ　りょういち）
情報通信システム開発部
グラフィックメータ，
ディスプレイECUの開発に
従事

中田　和行
（なかた　かずゆき）
情報通信システム開発部
グラフィックメータ，
ディスプレイECUの開発に
従事

梶田　健太郎
（かじた　けんたろう）
情報通信システム開発部
グラフィックメータ，
ディスプレイECUの開発に
従事

松本　壮樹
（まつもと　そうじゅ）
株式会社スリーディー
3Dグラフィックス技術の研究
に従事

石川　　巧
（いしかわ　たくみ）
株式会社スリーディー
3Dグラフィックス技術の研究に
従事

渡良井　葉麻
（わたらい　ようま）
株式会社スリーディー
代表取締役
3Dグラフィックス技術を
コアにした事業を展開