次世代携帯端末・移動体通信における

高速３ＤＣＧエンジンの開発

Development of a high-speed 3D indication engine in next generation portable terminal / mobile communication

酒井　雅裕[1)]　　伊藤　和彦[2)]
Masahiro Sakai　Kazuhiko Itou

1)　株式会社マイクロネット（〒064-8633 北海道札幌市中央区南１０条西１５丁目１－８ ビッグビル３Ｆ　E-mail: masa@sweetness.com ）
2)株式会社マイクロネット（〒064-8633 北海道札幌市中央区南１０条西１５丁目１－８ ビッグビル３Ｆ ）

ABSTRACT. Development of a high-speed 3D indication engine in next generation portable terminal / mobile communication. There is an element computer technology - 3DCG display technology for mobile computer entertainment in the future. This research and development purpose is development to the engine which handle 3DCG on the mobile terminal and create advantage for now technology. We implemented two engines of server side rendering and client side rendering. They were commercialized finally. For example, http://www.kensakun.net/I-mode/

１．背景

電気通信事業者協会の２００１年４月の発表によれば、２００１年３月までの携帯電話の契約者数は、６０９４万３４００人とされ、６０００万人を越えた。現在も１ヶ月で約１５０万人が増加し、携帯端末は堅調に普及傾向にある。モバイルコンピューティングコンソシアムの市場規模予測によれば、２００４年度には７８２０万人、移動体全体での規模は８４６０万人、対人口普及率６６．８％に達するとまで予測されている。

ＦＯＭＡ端末の３８４ｋｂｐｓダウンロードに代表される通信速度の飛躍的な改善や、携帯端末そのものの処理速度・表示速度の改善、何よりも移動体自体の普及率の高さを背景にして、良質な携帯電話サービスの提供が求められ、従来の音楽、映画を含む画像コンテンツの配信や、ゲームなどのコンピュータエンタテイメントの配信もかなりの需要が予測できる。

２．目的

将来必ず移動体上で必要となるコンピュータエンタテイメント配信の基盤要素技術に３ＤＣＧ表示技術がある。今回の研究開発の目的は、次世代携帯端末・移動体端末上で高速に３ＤＣＧを扱うエンジンを総合的に開発することでありアドバンテージを創出する事を目的とした。

３．実装の方向性

３ＤＣＧを配信する場合の問題点は大きく以下の３点に集約される。

１）表示スピードに耐えるテクスチャを含む、３Ｄオブジェクト構造の策定
２）通信速度･端末処理速度に見合ったデータ圧縮アルゴリズム
３）表示速度を優先した高画質レンダリングエンジンの選定

これら３点をクリアして初めて、高速且つ軽快な３ＤＣＧ表示を可能にすると考えられる。

また、携帯端末上でレンダリングエンジンを持つクライアントサイドレンダリングとサーバ側にレンダリングエンジンを持ちレンダリング結果のみ配信するサーバサイドエンジンの双方を検討する。何故なら現状の移動体端末のハードウエアリソースはかなり制限を受けている。従って数万ポリゴンのデータやテクスチャを配置したデータについては移動体端末上ですべて扱うことは事実上不可能で今回サーバサイドレンダリングを検討するに至った。

４．F/S

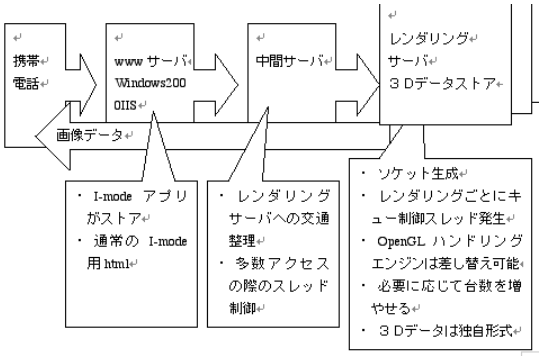
F/Sは２点について行った。この終了時点で、当初目標にしていたサーバサイドレンダリングとクライアントサイドレンダリングの双方のスペックを決定できた。

（１）クライアントレンダリングにおいては携帯端末上の実装言語の調査と、実装の可能性について追求した。この事業を開始した当時は、携帯端末上で無料かつ情報量が豊富な開発環境はｉ-modeの開発環境しかなく選択の余地がなかった。ｉ-modeのリソースはcode10KB、スクラッチパッド5KBで非常にリソースが少ないため、

レンダリングエンジンを Java3D に似せたライブラリ化する構想は断念せざるを得なかった。
（２）サーバサイドレンダリングの実装スペックを各種サーバの調査・言語の調査を行い、Windows2000server－IIS を基本システムとし開発言語は MS-VisualC++が妥当であると決定した。
別候補
Linux-GCC-C++または C によるライブラリと perl による呼び出し機構
クライアント側は html+JAVA アプレット

F/S の結果、高速レンダリング用のビデオカードのハンドリングを Linux からできないケースが多い。ハンドリングの方法は C++または C によるインターフェイス→OpenGL インターフェイス→ビデオカードドライバ→ハードウエアの手順となるが Linux 対応ドライバを持つ高速ビデオカードの入手が困難で開発可能性を断念した。

５．サーバサイドレンダリング
テスト環境
Pentium-II400MHz　GeForce3 Windows2000server
建物数１万　総ポリゴン数８万　都市データ
テクスチャ無制限
１００ｆｐｓ＋　２５６ＧＩＦ　インターレス
ハードウエアレンダリングファーム　OpenGL
図＿１



サーバサイドレンダリング図

図＿２

サーバサイドレンダリングの画像

（１）各サーバの役割
レンダリングサーバはクライアントからの情報を元にした３Ｄ空間上の視点での画像を生成するのが主な役割である。
クライアントから発信されたデータは中間サーバでパケットの交通整理がなされた後、画像生成に必要なデータが中間サーバからレンダリングサーバに送られる。
レンダリングサーバは中間サーバのリクエストに応じたメッセージデータや画像データを中間サーバに送り返す。
メッセージデータとは主に空間の位置情報を文字で表したもので、視点基準のキャプションが位置情報として示される。視点基準は、地図なら「南１０条西１５丁目交差点」、住宅間取り図なら「リビング」で、クライアントが位置選択を行う際の出発地点としての用途に用いられる。
これら視点基準データは、データ制作者にしか解らないのでレンダリングサーバがこれらをクライアントに配信できる仕組みになっている。

中間サーバはクライアントの UI 用の制御を行うのが主な役割であり、同時に「データの分類」、つまりレンダリングサーバとのデータ交換を円滑に行うための制御も範疇になる。
レンダリングサーバ自体レンダリングは高速に行えるが、一方でデータの読み替えが発生する場合は、高速な応答ができなくなる。あるクライアントが地図データを要求し、別のクライアントが住宅間取り図を要求する場合、一つのレンダリングサーバでこれを捌くとなると、データの読み替えをしなくてはならなくなる。これでは、多数のクライアントから同時刻帯にアクセスが集中した場合、実用的な範囲での応答速度が得られなくなる可能性が高くなる。
この問題を解決する為、レンダリングサーバは基本的に一つのコンテンツについて集中的に解決し、複数の別のコンテンツを扱う場合には、レンダリングサーバを分散する手法を取る。その為、中間サーバはそういったコンテンツ・データ毎のレンダリングサーバ選択というデータの交通整理も範疇になる。
なお、レンダリングサーバはアプリ単位なので、リソースが許す限り一つのマシンに複数のレンダリングサーバを待機させておく事は可能である。

中間サーバとレンダリングサーバの交信はソケットを使用し、ダイレクトに通信を行う。

（２）動作
今回のテストではレンダリングサーバを PentiumⅡ 400Mhz + GeForce3（又は G400）、Windows2000 マシン上で動作させた。
レンダリングサーバは基本的に一般的な HTTP サーバに似た動作をする。中間サーバからのリクエストをアクセプトした後、速やかに新たな送信スレッドを生成し専用のソケットで中間サーバと交信を行う。
中間サーバとのシェイクハンドは速やかに行う必要があるが、一方、レンダリングをマルチスレッドで行う訳にはいかない。グラフィックアクセラレータのハードウェアリソースはマシンに唯一つのものであるため、一般的なアクセラレータにてマルチスレッドで同時アクセスを行うと、内部的なレンダリングステートの変更が頻発し、かえってパフォーマンスを悪くする原因となる。
そこで、各レンダリングサーバはレンダリングスレッドを１つ生成し、中間サーバとのシェイクハンドを速やかに行ったあと、リクエストデータをレンダリングキューとして一時バッファにスタックしておく。
レンダリングスレッドはスタックの最下層からキューを取り出し、情報に基づいた画像生成を行い、キューを生成したスレッドに画像終了を通知する。

その後、そのスレッドは中間サーバへデータの返信を行い、交信を終了する（スレッドを閉じる）。

今回テストで用いたデータ（約８万ポリゴン）と先述のマシンスペック（+GeForce3）において,レンダリングスレッドは後述の携帯向けポストフィルタを通しても、100FPS 以上の画像生成が可能であるため、キュースタック～画像生成まではほぼ一瞬で終えることが可能である。つまり、レンダリングサーバのほとんどの時間はデータ通信が占める事になる。

ポストフィルタは、減色、（必要があれば）ディザリング、画像フォーマット変換（GIF）の３つの処理から成るが、携帯用途の画像サイズであれば、瞬時に計算を終えられる。ただし、この過程でファイル I/O 処理が入ることと、アクセラレータが効かない範囲なので画像解像度が大きくなると直ちに過負荷になることは予想される。この様な高解像度画像処理が必要な場合の運用では、マシンスペックの向上、物理的な分散数（マシンを増設する）の向上等、物理的なスペックを引き上げる方向性で検討する必要がある。

（３）レンダラに対する評価

今回のテスト運用では、マイクロネット社の全面協力でレンダリングシステムをベースに構築することが可能となった。

OpenGL を通じたハードウェアレンダラで、データ構造は独自構造のものである。この独自データ構造はテキストベースのフォーマットで冗長性、可読性が高い反面、読み込み速度が遅いという欠点がある。

レンダラの目的は、要求された画像を高速に生成することである。今回のテストケースで用いたレンダラの欠点、すなわち冗長なテキスト型のデータ読み込み速度が極めて遅いと言うことは、場合によっては致命的欠点である。３Ｄ空間データを読み替える様な用途（たとえば、住宅間取り図が多数あって、クライアント希望の間取り図を表示する様なケース）では、そのデータの読み込み速度がネックとなり実用的な扱われ方をしない可能性がある。

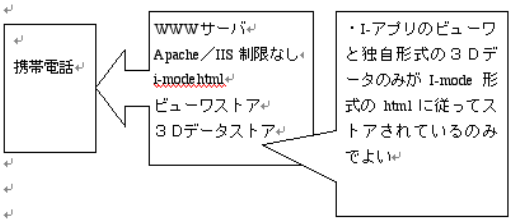
しかし、レンダラは先述の様に、基本的な中間サーバとの交信部分とは完全に切り離れた独立スレッドで動作しているので、レンダラ自体を用途に合わせて差し替える事は難しくはない。

コンテンツとデータの仕様が完全に固まった段階でメモリイメージに近いバイナリフォーマット＋データベース＋データキャッシュを定義できれば、局所データを多数切り替えて応答する必要があるコンテンツにも十分対応可能であると思われる。

６．クライアントサイドレンダリング

i-mode 側ですべてレンダリング
生成画像２５６ポリゴン　シェーディングベース
都度読み込み可能なので事実上ポリゴン数無限
カラー無制限
アニメーション可能　回転・移動シークエンス包含
全コードサイズ 7KB
表示スピードは 10fps
ポリゴンデータ→特殊形式のトランスレータ作成

図＿３



図＿４

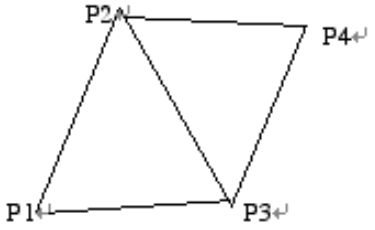
クライアントサイドレンダリング画像

（１） クライアントサイドデータ方式

クライアントレンダリングのデータシステムについてソフトウエア特許申請を準備中である。携帯電話のようにごく限られたリソースしか持ち得ない場合に通常のベクトルデータを持つには不合理な点が多く、以下のロジックを基本としてデータの軽量化を図った。

平面を三角パッチの集合体と考えた場合、２つのパッチの連続はベクトルデータの場合は p1(x1,y1,z1)-p2(x2,y2,z2,)-p3(x3,y3,z3):p2(x2,y2,z2,)-p4(x4,y4,z4)-p3(x3,y3,z3) などデータ記述が冗長になる。ポリゴンの場合それが更に連続することになる。

図＿５



三角ポリゴンの模式図

このような場合、端点の繋がり関係をN字形にトレースし、それを、一定のグループとして記述する方法 p1 p2 p3 p4:(x1,y1,z1)(x2,y2,z2,)(x3,y3,z3)(x4,y4,z4)と記述した方がデータ自体の冗長性はなくなる。もちろん画像構成エンジン側でこの並びを元にポリゴンを生成し直す必要がある。この概念を適応・改良して携帯端末上の独自ロジックを生成し、クライアントレンダリングの方式を確立した。

レンダリング方式は現在の実装した携帯端末のアプリケーションコード上限が 10KB であったことから通常の

ポリゴンシェーディングのみが限界であった。

７．まとめ

定めた目標

- 表示スピードに耐えるテクスチャを含む、３Ｄオブジェクト構造の策定
- 通信速度・端末処理速度に見合ったデータ圧縮アルゴリズム
- 表示速度を優先した高画質レンダリングエンジンの選定

と結果を比すれば、概ね達成できているように思う。

とりわけクライアントレンダリングについては、データ圧縮アルゴリズムの策定や7KBのコード量ですんでいる点、またある程度のスピードが確保できている点など評価できる到達点を確保できている。

ただ、サーバサイドではテクスチャリングについて無制限で可能であるものの、２００２年２月現在でテストに使用した携帯端末では、通信速度がスピードがあまりにも遅く(9600bps)、実用に耐えられない。現状の解決策としては、クライアントレンダリングを工夫しテクスチャリングを行うか、更に高画質エンジンをめざしてブラシアップする必要がある。

また、今回のテーマと直接関わりはないが、3DCGの普及にはやはり簡単でコストのかからない3Dモデリングの方法論の確立が不可欠であると痛感した。この点をクリアできなければどんなにすばらしい3Dエンジンを作成しようとも普及は困難である。

８．参加企業及び機関

株式会社マイクロネット

９．参考文献

[1]Jackie Neider,, Tom Davis, Mason Woo: OpenGL programming Guide, Addison-Wesley, 1993

[2]安藤幸央、他：OpenGL プログラミング、ＮＫエクサ、1995

[3]Clayton Walnum:3-D Graphics Programming with OpebGL, Prentice Hall, 1996

[4]山口富士夫：実践コンピュータグラフィックス　基礎手続きと応用、日刊工業新聞社、1987