マルチディスプレイを用いた高解像度プレゼンテーション環境の構築

千葉 豪 † 柴田 義孝 †
† 岩手県立大学大学院 ソフトウェア情報学研究科

1 はじめに

近年、ブロードバンドネットワークサービスの普及により IP ネットワーク上で DV や HDV のような高精細映像の配信や, 高性能ビデオカードを搭載した複数の PC と複数ディスプレイやプロジェクタを組み合わせたレンダリングクラスタの構築がソフトウェアレベルで可能となり, 高精細で大画面かつ高臨場感を持つディスプレイ環境が実現できるようになった [1] [2].

そこで、本研究では、大規模高解像度によるバーチャルリアリティや 2, 3 次元映像による協調作業を実現するため、複数 PC クラスタと液晶ディスプレイを組み合わせ、これらを高速ネットワークで接続することにより、実質的にクラスタサイズに比例した高解像度を達成できるプレゼンテーションシステムを開発する。

本システムはアプリケーションと独立してミドルウェアとして実装し、ピクセルデータのストリーミング機能やディスプレイ間同期表示機能や多地点間映像表示機能を実現し、高速ネットワーク上で柔軟に制御を可能とする。

本稿では、本システムのアーキテクチャや実現方法およびプロトタイプシステムについて述べる。

2 タイルドディスプレイシステム

本研究で用いる出力装置は複数液晶ディスプレイを組み合わせたタイルドディスプレイを用いる. これは一般的な LCD ,DLP 等のディスプレイを複数配置し, 仮想的に高解像度ディスプレイ環境を実現するもので, プロトタイプシステムでは 1600x1200 の解像度の LCD を 4〜16 台組み合わせることにより, 3200x2400 〜6400x4800 といった高解像度の映像出力を実現する. 各タイルにはレンダリングを行うための複数 PC によるクラスタを構成し, クライアントから受け取ったピクセルデータをタイルに対応した計算機がレンダリング処理を行う.

これにより大型ディスプレイのような専用のハードウェア用いることなく安価な PC によるシステム構成で高臨場感のある表示が可能となり, 表示される映像も大画面でかつ, 高精細で表示できるので, 現実に近いサイズでの協業作業, 複数地点間でのコミュニケーションが実現できる.

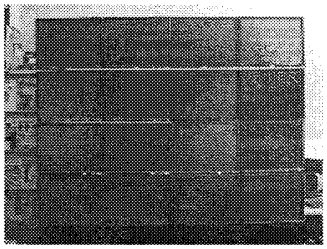

図 1: タイルドディスプレイ

High-Resolution Presentation Environment with a Multi Displays

† Go Chiba(g231f012@edu.soft.iwate-pu.ac.jp)
† Yoshitaka Shibata(shibata@iwate-pu.ac.jp)
Graduate School of Software and Information Science, Iwate Prefectural University (†)

2.1 アーキテクチャ

本システムのアーキテクチャを図 2 に示す. 出力装置として前述したタイルドディスプレイを用いることを想定しており, 各レイヤーの機能は下記のようになる.

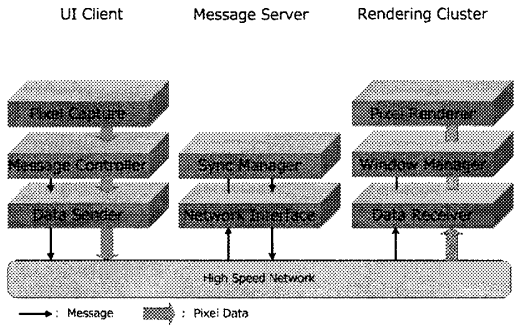

図 2: システムアーキテクチャ

2.2 UI Client

UI Client ではアプリケーションを提供すると同時に , Pixel Capture モジュールがピクセルデータをグラフィックスカードのフレームバッファから取得し Send Buffer へとデータが格納される. その後 Data Sender が Send Buffer 内のデータを レンダリングクラスタへピクセルストリームとして送信を行う. また, Message Controller モジュールが送受信の同期のためのメッセージを生成し, Data Sender モジュールを介して Message Server へ同期信号を送信する.

2.3 Message Server

Message Server では UI Client, Rendering Clusters から受け取った同期信号をもとにタイルドディスプレイへの描画処理の同期を計り, その後 Rendering Clusters へ描画用の信号を発信する. これにより各タイル間の同期をとり, スム ーズな描画処理が行われる.

2.4 Rendering Cluster

Rendering Clusters では UI Client から送信されたピクセルストリームを受けた後それらのデータを Recv Buffer へと格納する. その後 Window Manager モジュールが Message Server からの同期信号を受け取ると Recv Buffer 内のデータを取得し展開した後イメージを担当するタイルに適合するようリサイズなどの処理を加え, タイルへの描画処理を行う.

3 システムの機能

3.1 ディスプレイ間同期

ディスプレイ間の同期はメッセージのやりとりによって行われ, そのタイミングは図 3 の示す通りとなる. UI Client は同期メッセージを Message Server へ送信後各クラスタへピクセルストリームの送信を開始する. 各クラスタはそれぞれピクセルデータを受け取ったら描画の準備ができたことを Message Server へと通知する. Message Server では全てのクラスタからの通知を受け取ったら, 各クラスタに更新メッセージを送信し, 描画処理を開始するよう命じる. また, それと同時に UI Client へ次のデータを送るよう通知をし, UI

Client は次のフレームのピクセルデータをストリーミングを開始する.

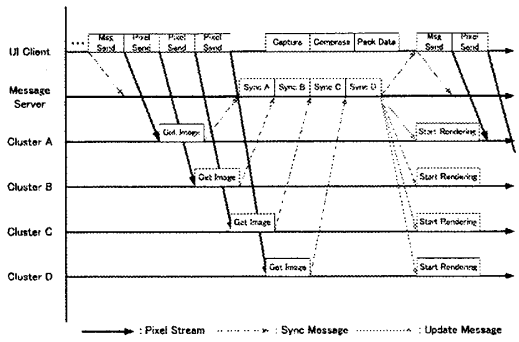

図 3: 同期フロー

3.2 ピクセルデータの取得

UI Client 上でのピクセルデータの取得には図 4 に示すとおり OpenGL の機能である *glReadPixels* 関数によって行い, 毎フレームをフロントバッファから RGB にて各ピクセル値を取得する. まず, *glReadBuffer* では読み込む先のフレームバッファの種類を指定する. その後, *glPixelStorei* にてバッファの読み取り方を 1 byte ずつ読み込むように指定し, *glReadPixels* によって読み取る領域の左下の座標 (*x,y*), 読み取る領域の幅と高さ (*width, height*), 取得する色情報の形式 (*GL_RGB*), 読み取ったデータを保存する配列の型 (*GL_UNSIGNED_BYTE*) を指定してフレームバッファからピクセルデータを取得する.

```
glReadBuffer(GL_FRONT)
glPixelStorei(GL_UNPACK_ALIGNMENT, 1)
pixeldata = glReadPixels(x,
                         y,
                         width,
                         height,
                         GL_RGB,
                         GL_UNSIGNED_BYTE)
                           .
                           .
                           .
```

図 4: キャプチャメソッド

3.3 データサイズ

ストリームとして扱うピクセルデータは 4800x3600 の解像度の場合

$$4800(width)*3600(height)*3(RGB) = 51,840,000bytes$$

となり, ピクセルデータを Frame Buffer から取得後, UI Client 上の Send Buffer へと格納される. その後, Data Sender モジュールが TCP プロトコルにより各クラスタへとストリーミングを開始する. 各クラスタでは Data Receiver モジュールが受け取ったデータを Recv Buffer へと格納し, Window Manager モジュールが展開処理やフレームバッファへの読み込みを行う. その後 Message Server からの更新メッセージを受け取ると Pixel Render へ働きかけ描画処理を行う.

4 プロトタイプシステム

プロトタイプシステムとして現時点において UI Client, Rendering Clusters の 2 つのコンポーネントを構築し実際に 4 面のタイルドディスプレイへの描画の確認を行った. また、比較評価のため SAGE[2] を用い 16 面ディスプレイへの VR アプリケーションの出力も行った。ここでは, VR アプリケーションの例として過去に筆者らが開発を行ったバーチャル伝統工芸システム [3] [4] への適用を行っている.

また, システムではタイルドディスプレイへの出力のためにバッファからのピクセルデータの取得, 分割, 配信といった基本的な機能を実装した.

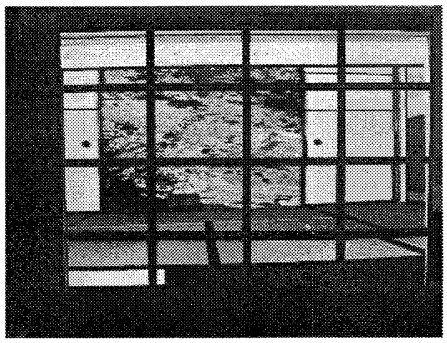

図 5: VR アプリケーションへの適用

5 評価手法

本システムのプロトタイプ完成後は"フレームレート", "利用帯域"に関して性能評価を行う. フレームレートはタイルドディスプレイにおいて描画を行うタイル数を変化させた際にフレームレートがどのように変化をするか評価をする. また利用帯域では UI Client, Message Server, Rendering Clusters の 3 地点間での利用帯域を測定する.

6 まとめ

本研究では複数ディスプレイを用いた高解像度プレゼンテーション環境を提案し, バーチャルリアリティアプリケーションへと適用した. これは専用のグラフィックスハードウェアを用いず, 安価なディスプレイを複数台利用することで容易に高解像度環境を実現し, これにより高精細な三次元仮想空間やビデオイメージの表現が可能になるといったことが期待される.

今後は更にプロトタイプシステムの構築を進め, 描画を行うタイル数の変化させた場合のフレームレートの変化, CAVE システムを利用した場合とタイルドディスプレイを利用した場合の没入感, 臨場感の違いを評価していく.

また, 現時点でのプロトタイプがユニキャスト通信を利用しているため利用帯域がタイル数に比例して増大することや, UI Client の負荷が大きいといった問題が存在している. そこで, マルチキャストを利用することで利用帯域を抑えられることが可能か, UI Client での負荷の軽減手法に関して今後検討していきたいと考えている.

参考文献

[1] Bruno Rafn, Luciano Soares, "PC Clusters for Virtual Reality", IEEE Virtual Reality Conference, 2006

[2] Byungil Jeong, Luc Renambot, Rajvikram Singh, Andrew Johnson, Jason Leigh, "High-Performance Scalable Graphics Architecture for High-Resolution Displays", Tech Paper,2005

[3] 杉田薫, 宮川明大, 柴田義孝, "JGN を利用した VR デジタル伝統工芸システム", 情報処理学会論文誌 Vol.43, No.2, 2002

[4] 石田智行, 宮川明大, 柴田義孝, "超高速ネットワークをベースとした没入型環境システムにおける共有法", 情報処理学会第 68 回全国大会 5T-11 pp.4-239〜240,2006