Vol.2014-HPC-143 No.28
2014/3/4

ユビキタスシミュレーションのための携帯端末を用いたタイルドディスプレイシステム

柴田　直樹[†1]　荒川 文貴[†1]　福間　慎治[†1]　森　眞一郎[†1]

概要：これに対して我々は，遠隔地のスーパーコンピュータ上での科学技術計算に対して対話的な操作を可能とする実時間遠隔シミュレーションステアリング, ならびにネットワーク接続可能な場所であればどこからでも大規模シミュレーションを対話的にステアリング可能とするユビキタスなシミュレーション・インタフェースの開発を行っている．本論文では，このようなユビキタスな対話型シミュレーションを実現するための入出力インタフェースとして携帯端末を用いた高精細タイルドディスプレイシステムの提案を行う．また，携帯端末のタッチインターフェースを用いてステアリングインターフェースを設計し，実装を行った結果を報告する．

キーワード：タイルドディスプレイ，ディスプレイウォール，タブレット PC，ユビキタスシミュレーション，

1. はじめに

近年の計算機性能の急速な向上に伴い，インタラクティブな実時間シミュレーションへの期待が高まっている．フライトシミュレーション [1] や航空管制シミュレーションのようにコンピュータ上のシミュレーション結果を操作者が直接体感し，その反応として対話的にシミュレーションをステアリング可能なシミュレーションの形態は”human-in-the-loop simulation” あるいは”interactive simulation” と呼ばれる．近年盛んに研究が行われている，災害時の緊急避難シミュレーションなどもその一例である．従来，このようなシミュレータ上で行われてきたシミュレーションは主に離散事象シミュレーションであったが，これをスーパーコンピュータ上の科学技術計算のシミュレーションにも拡張する試みも進められている [2,3,4]．しかしながら，実時間での対話的なステアリングまでを考慮した研究は始まったばかりである [5,6]．これに対して我々は，遠隔地のスーパーコンピュータ上での科学技術計算に対して対話的な操作を可能とする実時間遠隔シミュレーションステアリング, ならびにネットワーク接続可能な場所であればどこからでも大規模シミュレーションを対話的にステアリング可能とするユビキタスなシミュレーション・インタフェースの開発を行っている．本論文では，このようなユビキタスな対話型シミュレーションを実現するための入出力インタフェースとして携帯端末を用いた高精細タイルドディスプレイシステムの提案を行う．また，携帯端末のタッチインターフェースを用いてステアリングインターフェースを設計し，実装を行った結果を報告する．

2. 研究背景

2.1 ユビキタスコンピューティング環境

組み込み向け CPU のマルチコア化, GPU の混載, 大容量メモリの搭載を初めとする携帯型端末の急速な性能向上に伴い，携帯端末は単なる情報検索・閲覧の入出力デバイスから，ユビキタスかつ強力なクライアントサイドコンピューティングデバイスに変わりつつある．また，ITU の IMT Advanced 規格の承認により数年後には携帯機器向けの通信速度は移動時においても 200Mbps の帯域を享受できる可能性がでてきたことも，携帯端末でのユビキタスな HPC 環境実現の可能性を後押ししている．一方で，携帯機器向けのプログラム開発環境においても，クライアントサイドでの計算資源の活用，ハードウェア構成や OS の異なるマルチプラットフォーム向けのプログラム開発の効率化を目的として HTML5 技術を活用した Web ベースのアプリケーション開発が急速に進んでいる．　本研究でも，遠隔地で実行中のシミュレーションをユビキタスにステアリングするための入出力インタフェースとして携帯端末のタッチインターフェースを用いるとともに，WebGL ならびに WebSocket を用いた Web ベースのステアリングイン

[†1] 現在，福井大学

タフェースの構築を行った.

2.2 タイルドディスプレイシステム

ユビキタスかつ対話的な HPC 環境の実現には，遠隔地で行われているシミュレーション結果の遠隔可視化，遠隔ステアリングが必要である．大規模シミュレーション結果の遠隔可視化の手段としては，通信バンド幅の確保，提示画像解像度の向上を目的としたタイルドディスプレイシステムが提案されている．タイルドディスプレイはディスプレイウォールと呼ばれることもあるが，本論文では一律にタイルドディスプレイと呼ぶ．タイルドディスプレイとは複数の表示端末をタイル状に配置し，1 つの大きなディスプレイとして使用することで超高解像度のディスプレイを実現できる技術である．タイルドディスプレイの実現方法にはマルチモニタ対応の GPU を用いた小規模なものや，壁一面にディスプレイを足りつけるなどクラスタベースのものがある．クラスタベースのタイルドディスプレイはコモディティ PC を LAN などで相互接続したものが知られており，コストが低く，パフォーマンスと解像度に拡張性がある．タイルドディスプレイシステムを実現するミドルウェアとしてはイリノイ大学で開発された SAGE や米国 Kitware 社の Paraview が有名である．しかしながら，その多くが遠隔地ではあるものの特定の場所に設置する大型の高精細可視化装置への応用である． 次に，遠隔ステアリングのためのステアリング入力について考えると，大型のタイルドディスプレイのコントローラと連携して動作する Wireless 3D マウスや赤外線カメラと連携したジェスチャ入力等，ステアリング入力のために別途設備が必要なものが多く固定型の大規模システム向けが殆どである． これに対して，本研究では高精細化が進む携帯型端末のディスプレイをタイル状に並べて実現するタイルドディスプレイと、携帯端末に本来備えられているタッチパネルを入力としたシミュレーションステアリングインタフェースの構築を行った.

2.3 ユビキタスシミュレーション環境の構築

今回システムの実装方法を検討する上で, 我々の重 要視する以下の 4 つの要件を定義した.

(1) システムを利用する上で特別な環境や知識を必 要としない.
(2) 携帯端末の OpenGL 規格である OpenGL ES を最大限活用できる.
(3) 対称かつ双方向の対話性を確保できる.
(4) サーバサイドコンピューティングとクライアントサイド GUI の連携が容易である.

これらの要件に対して, 代表的な既存技術について検討調査を行った．調査の結果, 本研究では Canvas または WebGL2) と WebSocket3) を併用した Web インタフェースを構築することとした. 以下, それぞれ概要について説明する.

2.3.1 Canvas および WebGL の概要

Canvas とは,Web ブラウザ上に図を描くために策定された HTML 要素である. HTML 上に Flash や Java のようにプラグインを使用せずとも,Javascript を用いて図やアニメーションを表現することができる. さらに、Canvas 要素をと Javascript を利用することで, ウェブブラウザ上で GPU によってアクセラレートされた 3 次元コンピュータグラフィックを表示させる標準仕様が WebGL である. OpenGL ES 2.0 が動作する環境で WebGL に対応したブラウザと GPU があれば, 特別なプラグインを必要とせずに WebGL アプリケーションが表示可能である. これらを Web ベース入出力インタフェースとして利用することで, 環境に依存しない汎用性の高い対話型アプリケーションの設計が可能となる.

2.3.2 WebSocket の概要

WebSocket とは, コンピュータネットワークの通信規格のことであり, サーバとクライアント内での双方向通信規格である. 従来の通信規格である XMLHttpRequest との大きな違いは, サーバ側からクライアント側にデータをプッシュできることと, 1 回の接続要求で, 任意の回数の双方向通信が可能であることである.

2.3.3 Web インタフェースの概要

サーバ側へのデータ入力及びサーバ側からの結果データの表示は Web ページから行い, データ通信には WebSocket を利用する. WebSocket を利用することで Web ページ上からサーバとの双方向通信が可能となり, クライアントがサーバへ送信依頼を送らずともサーバが一方的に結果データをプッシュこれらの方法を利用することで,Web ページからの対話型遠隔シミュレーションを可能にするシステムを実現した. システムの流れは, (1) シミュレーションに対する処理データを対応ページに入力する.(2) 入力されたデータを WebSocket を利用してサーバに送信し, シミュレーション実行スクリプトに渡す.(3)Python で記述されたシミュレーション実行スクリプトから C 言語で記述されたシミュレーションコアを呼び出し, シミュレーションパラメータを変更する. (4) シミュレーション実行スクリプトから結果を取り出す. (5)WebSocket を利用して対応ページへ結果データを送信する. (6)Canvas または WebGL を用いて結果データを反映させる.Web ページでの表示処理はクライアント側が負担する． することが可能になる. これによりサーバ上でのシミュレーションの進行に追従した結果表示が可能となる. サーバ上でのシミュレーション自体の処理と Web ベースインタフェースの処理は並列かつ非同期的に実行される。そのため、クライアントからのインタラクション がない場合にも、最後に与えられたシミュレーションパラメータ (境界条件等) に従ってシミュレー

ションは継続し、一定間隔で計算結果はクライアイント側に送出され, Web ブラウザに表示されている画像が更新される。

GPU が無い場合の Canvas,GPU がある場合の We-bGL の両方に対応する Web ベースインタフェースを検討し, 対話型遠隔シミュレーションを実装した. また, 今回は GPU を用いた処理は画像の表示のみを対象としているが,WebGL を利用して GPU 自体にシミュレーション等の計算をさせることで, Web ページ上から特別な環境や知識を必要とせずに手軽にシミュレーションをすることが可能なのではないかと考えた.

2.3.4 マルチクライアント

WebSocket を用いた双方向通信では Web サーバ側のプロセスは, クライアントの間で Socket を確立しつづける. そのため, クライアント毎に Web サーバ側でサーバプロセスを起動しなければならない. この仕組 みは,Apache の pywebsocket モジュールを用いるこ とで実現した . シミュレーションサーバはマルチクライアントに対 応するために, どのクライアントから入力があっても, パラメータを変更できるようにする.select 関数は, 複 数のソケットを監視することも可能であり, ソケットの変化があれば, それに合わせた処理を行うことができる. 現時点では複数クライアントからの同時入力に は対応しておらず、仮に同時期にパラメータ変更が発 生した場合にはそれらを逐次化して順次変更があった ものとして処理を行っている。シミュレーション結果データの配信に関しては, デー タを代表として受け取るプロセスを Web サーバ上に 作成することで, シミュレーションサーバと Web サー バ間の通信回数を 1 回に削減した. 代表受信プロセス のデータは,Web サーバ内でそれぞれのサーバプロセ スに分配した後, 各々のクライアントに配信する. 現在は、シミュレーションサーバと Web サーバ間ならびに Web サーバとクライアント間で授受するデータの解像度は同じであるが、今後は利用かのうな通信帯域を考慮した解像度制御等も検討していく予定である。

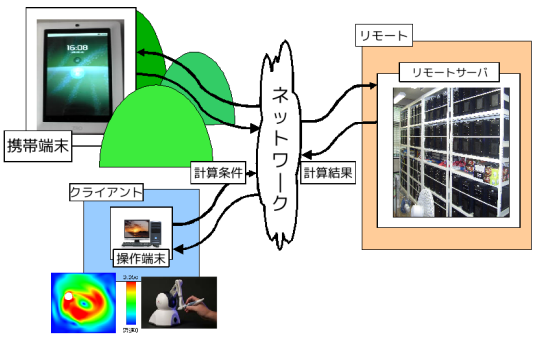

図 1 対話型遠隔シミュレーション

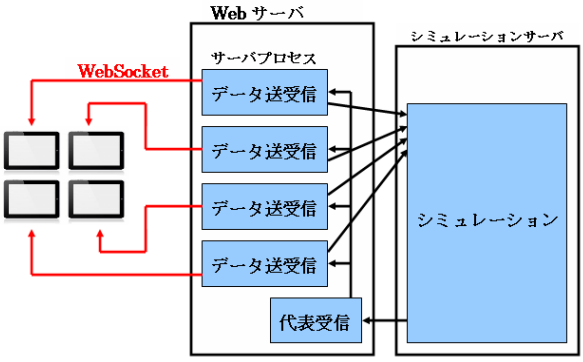

図 2 システム構成

3. タイルドディスプレイシステム

今回のシステムは複数の携帯端末を用いてタイル状に並べ, 一つの画面を表示する. この時, 携帯端末側からタッチインターフェースによる操作が可能である. 今回はユビキタスシミュレーションをタイルドディスプレイシステムで実装する.

3.1 携帯端末を用いたタイルドディスプレイの実現

3.1.1 ディスプレイ分割および画面表示

タイルドディスプレイを構成する各クライアントがディスプレイのどの領域を担当するかを決定する処理とその決定に応じた画面分割処理が必要である. 今回の実装においては, クライアント端末の構成台数を 4 台に限定し, ディスプレイ割り当てはクライアント端末からの接続要求順に静的に同一サイズの領域を時計回りに割り当てる方式を採用し [3.1.1], 画面分割はサーバ上で行った. 携帯端末を操作端末に用いることを考えると画面サイズをそれに合わせる必要がある. 接続した時点でクライアントの解像度等を入手し, 動的に調整することが理想的ではあるが, 今回は簡単のため解像度を使用している携帯端末に調整、固定した.

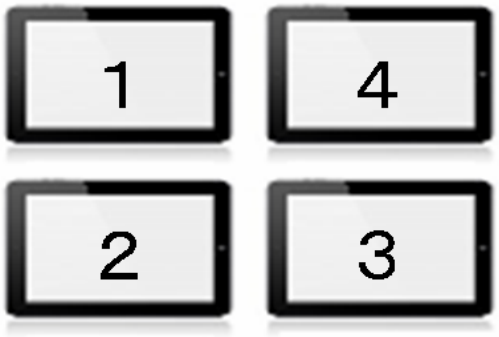

図 3 ディスプレイ割り当て

築する上で必要となる実装上の課題を本項で説明する。

3.1.2 クライアント同期

複数クライアント端末でタイルドディスプレイを構成する場合，同一時刻のシミュレーション結果を各クライアント端末に分割して表示しなければばらない．厳密にはクライアント端末上でのフレーム切替の同期が必要であるが，実装上のオーバヘッドが大きい．そのため，本実装ではWeb サーバ側での同期処理で許容することとした．さらに実装の容易さを考え，代理受信プロセスでの配信処理の逐次化により簡易なフレーム同期を実装した．

3.2 タッチパネルを用いたステアリングインターフェース

3.2.1 仮想タッチパネル

大きな仮想的なタッチパネルを領域分割して各携帯端末のスクリーンに割り当てる。Web ページとしてディスプレイインタフェースを実現したことの利点として、ページ内にステアリングのためのメニュー表示が可能となった。端末 ID と端末内の座標から仮想タッチパネルへ座標変換して仮想タッチパネルへのステアリング入力として処理する。

3.2.2 ステアリング情報

各携帯端末のタッチパネルからのステアリング情報は [mode,x,y] という int 型配列として送信される。この情報は結果送信プロセス内とシミュレーションプロセス内で認識が可能であり、ステアリングの拡張性に優れている。シミュレーションのステアリング、色情報の変更、画像の変更等の操作もこのステアリング情報によって実装が可能である。以下ではいくつかのステアリング案の提案と実際に実装したステアリングについて解説する。

3.2.2.1 ステアリング案

ステアリング情報は mode の値によって以降のデータの種類を宣言する。例えば座標情報が入っているとしても、mode の値によってそれがシミュレーションのステアリングなのか、画像の拡大縮小等の画像へのステアリングなのかを使い分けることが可能となる．その他にも機能拡張は多数考えられるが現在はその実験段階として二つの実装を完了している。

3.2.2.2 拡散係数の変更

現在実装されているシミュレーションは熱拡散シミュレーションである．シミュレーション内容は容易に変更が可能だが，クライアントによるシミュレーションのステアリングという意味では入力座標以外の情報を送信することは重要である．そこで今回は熱拡散シミュレーションの熱拡散係数をクライアントが段階的に変更できるよう実装した (図 4)．これにより，座標送信以外の情報によるシミュレーションのステアリングが実現可能であることが実証できた．

3.2.2.3 初期化

また，拡散係数の変更の他にクライアントからの初期化を可能とした．ただし，どちらも HTML 上に設置したボタン操作による干渉であり，canvas 外に設置しているため，canvas に内蔵する形への改良が考えられる．

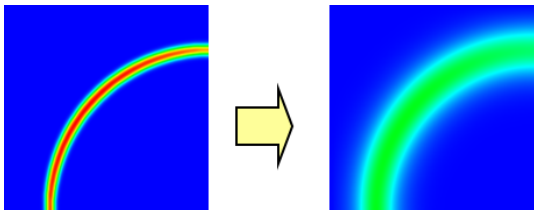

図 4 拡散係数の変更

3.2.3 座標修正

実装当初のシステムでは，並べられたクライアントのうち一つを操作することでシミュレーション全体に操作を反映していた．しかしながら，これは直感的な操作という意図に反し，操作する上でシステムの仕様に対する知識が必要となってしまう．操作端末内で座標を修正することも可能だが，シミュレーションサーバ側でクライアントを認識しクライアントに割り当てられた画面位置に応じて座標を修正した．

4. 実装および計測

4.1 実装状況

現在使用している携帯端末の短辺解像度が 800 程度であり，4 台の携帯端末を使用して欠損無くシミュレーションを表示できるシミュレーションサイズは 1024*1024 である．そこで近い将来利用可能となる携帯端末の性能を考慮し、目標最高解像度を 2048*2048 とした．現在実機での動作確認が完了しているのは 512*512 でのタイルドディスプレイである．1024*1024 及び 2048*2048 のシミュレーションに置いては今のところ 4 つの操作端末のうち 1 つのみ表示可能で，4 台での動作検証が課題となる．今回実験可能な解像度の画像は端末 1 台での表示が可能なため、全結果を 1 画面で実行した場合とタイルドディスプレイシステムに変更した場合に起こる速度変化についてシミュレーションサイズ 512*512 及び 64*64 のサイズで検証実験を行った．実験に使用したアプリケーションは熱拡散シミュレーションで，今回はシミュレーションサーバ上でのプロセスと Web サーバ上での分割・分配を担当する代表受信プロセス，そして操作端末での動作速度を 1 画面表示時とタイルドディスプレイ表示した場合について観測する．

4.2 システム構成

本システムでは 2 つのサーバ機を用いて実装を行った (図 5)．シミュレーションサーバでシュレーション計算を行い，それを Web サーバにソケットで送信する。この Web サーバ上では二種類のプロセスが動作している．一つは結果を分割，配布している代表受信プロセス．もう一つはクライアントと直接 websocket でつながっているデータ送受

信プロセスである．データ送受信プロセスでは代表受信から送られてきた画像データを操作端末に送信する．このプロセス内では受信頻度に関わらず送信頻度は一定に設定可能で，操作端末とのリンク速度を考慮した送信間隔を設定できる。このプロセスはそれ以外にクライアントから送信されたステアリング情報をシミュレーションサーバに送信する役割を担う．ここでは代表受信プロセスを介さず直接シミュレーションサーバに接続，送信する．操作端末はあらかじめ決められたソケット番号に接続することで Web サーバとの websocket 通信を確立する．

シミュレーションサーバでは、ステアリング情報があった場合はそれを用いて (図 6 の 1)，無かった場合はこれまでの情報を用いてシミュレーション計算を行う (図 6 の 2)．シミュレーション計算はここで可視化され (図 6 の 3)，シミュレーションプロセスで計算されたシミュレーション結果は Web サーバ内の代表受信プロセスへ送信される (図 6 の 4)(図 6 の 5)．代表受信プロセスでは受信したシミュレーション結果分割し (図 6 の 6)，クライアントに接続されている結果送信プロセスへと分割後の画像を適切に送信 (図 6 の 7)，これをクライアントの数だけ繰り返す (図 6 の 8)．結果送信プロセスでは受信した画像をクライアントに送信する (図 6 の 9)(図 6 の 10)．これがクライアントで受信されるとクライアントはテクスチャを更新し，画像を切り替え、再描画を行う (図 6 の 11)(図 6 の 12)(図 6 の 13)．このときタッチ等の入力が検出されるとステアリング出力を行い受信待ちに戻る (図 6 の 14)(図 6 の 15)．クライアントから送信されたステアリング情報は，まず結果送信プロセスに送信される (図 6 の 16)．結果送信プロセスでは受信したステアリングデータを認識し，シミュレーションサーバにステアリング情報を送信する (図 6 の 17)(図 6 の 18)．シミュレーションサーバは受け取ったステアリング情報を用いてしシミュレーション結果を再計算する (図 6 の 1)．結果送信プロセスにおいて，シミュレーション結果の送信 (図 6 の 9)(図 6 の 10) とステアリング情報の送信 (図 6 の 16)(図 6 の 17)(図 6 の 18) はそれぞれ独立しており，ステアリング情報が入力されていなくてもシミュレーション結果の送信は行われる．

4.3 実験

今回はシミュレーションサーバ上でのプロセスと Web サーバ上での分割・分配を担当する代表受信プロセス，そして操作端末での動作速度を 1 画面表示時とタイルドタッチパネル表示した場合について観測する．実験は熱拡散シミュレーションを動作し，この際フレームレートに制限はかけず，すべてのプロセスがソケットによる同期通信での停止を除いて可能な限りの速度で動作する．代表受信プロセスにおいて，all send は受け取った結果を分割し 4 つのデータを分割及び送信する全ての時間 (図 6 の 8)，send 1 はそのうち 1 つを送信する時間 (図 6 の 7)，cut はうち 1 つを分割する時間 (図 6 の 6)，all はシミュレーション結果の処理 1 回にかかる時間である (図 6 の 5,6,7,8)．シミュレーションサーバにおいて send とは計算した結果を代表受信プロセスに送信する時間 (図 6 の 4)，sim とはシミュレーション計算にかかる時間 (図 6 の 2,3)，all とはシミュレーション結果の処理 1 回にかかる時間である (図 6 の 1,2,3,4)．クライアントにおいて receive とは結果画像を受信する時間 (図 6 の 11)，draw とは 1 回の画面描画にかかる時間 (図 6 の 12,13)，all とは画面更新 1 回にかかる全ての時間 (図 6 の 11,12,13,14,15) である．1 画面で一つのクライアントで表示した場合を 1 画面とし，タイルドタッチパネルシステムで計測したデータをタイルドとして表 2，表 3 に記した．

表 1 実験環境

クライアント	vitro-tabletPC4 台
ディスプレイ	7 型 TFT 液晶　マルチタッチディスプレイ 1280*800 ドット
CPU	Amologic8726-MXs (Cortex A9 Dual Core)1.2GHz
メモリ	1GB
無線 LAN	IEEE802.11g

Web サーバ	
CPU	Intel Core i7-860 Processor (8M Cache, 2.80 GHz)
メモリ	4GB

シミュレーションサーバ	
CPU	Intel Core2 Duo Processor E6700 (4M Cache, 2.66 GHz, 1066 MHz FSB)
メモリ	2.0GB

4.4 実験結果

シミュレーションサイズを 512*512 にして計測した結果を表 1 に，64*64 にして計測した結果を表 2 に示す．タイルド表示した場合，全体を 1 画面に表示したときと比べると代表受信プロセス及びシミュレーションサーバの更新は遅くなっている．しかし，操作端末側での画面更新時間は短かくなっている．

表 2 シミュレーションサイズ 512*512 での実験結果

代表受信プロセス [msec]					
	all send	send 1	cut	receive	all
1 画面	0.501	0.479	-	28.174	28.708
4 分割	6.943	0.133	1.951	24.695	31.738

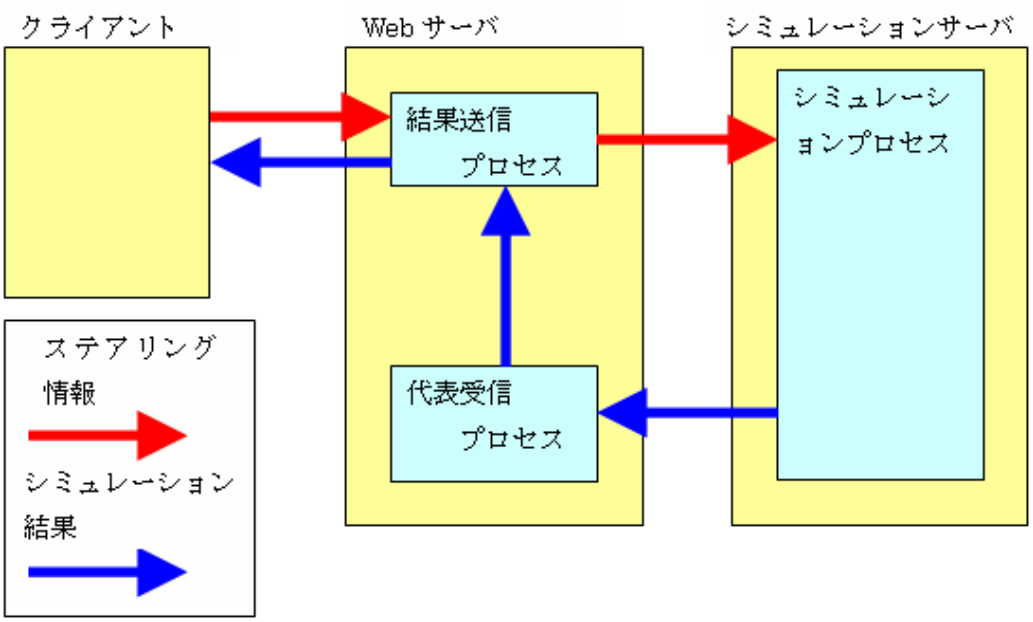

図 5 システムの流れ

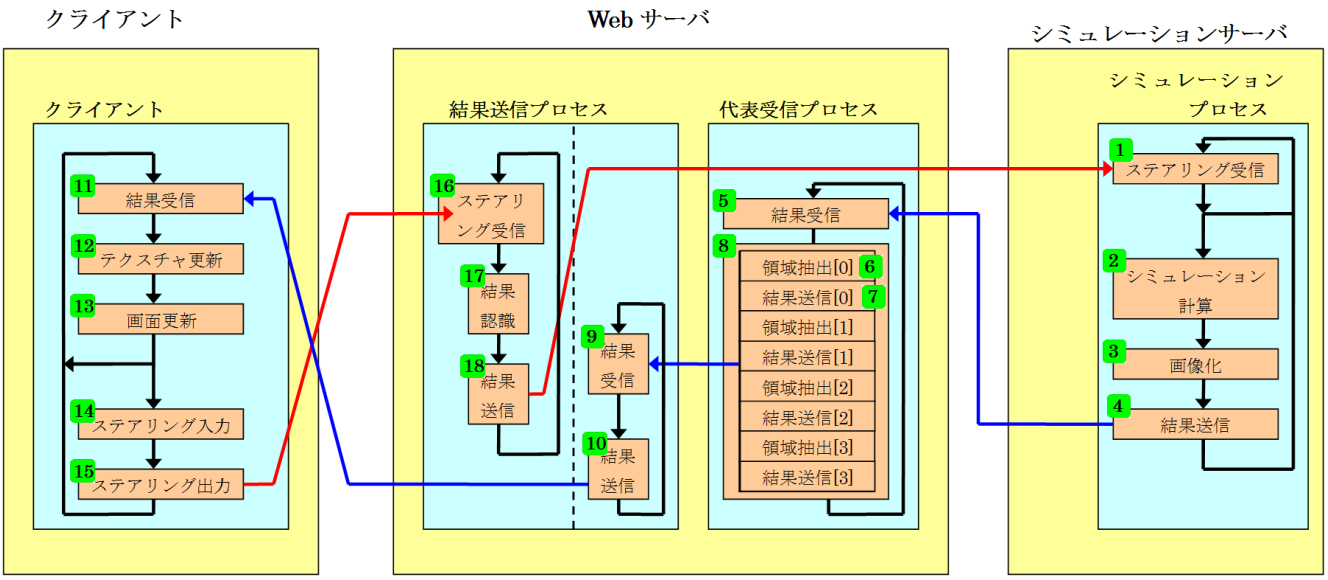

図 6 システム詳細

シミュレーションサーバ [msec]			
	send	sim	all
1 画面	3.919	24.730	28.670
4 分割	8.299	22.622	31.638

操作端末 [msec]			
	receive	draw	all
1 画面	677	1.05	762
4 分割	250	1.00	566

表 3 シミュレーションサイズ 64*64 での実験結果

代表受信プロセス [usec]					
	allsend	send 1	cut	receive	all
1 画面	43.2	26.0	-	417	362
4 分割	166	19.8	34.4	428	231

シミュレーションサーバ [usec]			
	send	sim	all
1 画面	19.9	390.8	419.5
4 分割	19.9	390.1	419.5

4.5 考察

まず全体のボトルネックとなっているのは操作端末であ

操作端末 [usec]			
	receive	draw	all
1 画面	11100	510	17700
4 分割	26000	1500	40000

る．Web サーバでは 1 回の画面更新に 30msec 程度かかっているのに対し，操作端末では 500 から 700msec 必要となる，4 分割表示した場合，全体を 1 画面で表示したときと比べるとサーバ側の画面更新は遅くなっている。これは Web サーバ上で画面分割を行っているため，その分割処理に時間を使っているためだと考えられる．しかし，操作端末側での画面更新時間は 762msec から 566msec へと短縮できたことがわかる．これはシミュレーション結果の受信にかかる時間と base64 形式で送信されたデータを元の形式に復元するための処理時間が短縮されたためだと考えられる．また，代表受信プロセスでの処理が増大したことで同期通信を行っているシミュレーションサーバの通信待ち時間が増加し，シミュレーションサーバの画面更新時間も増加している．実験の結果，サーバ側の処理時間が増加した代わりに操作端末側の処理時間は短縮された．しかし，ボトルネックとなっているのが操作端末側であることから，全体の画面更新時間は短縮できたと言える．合計で送信するデータサイズはどちらも同一だが，携帯端末の無線通信速度が無線ルータの通信速度を下回っているため端末を分割することでデータ通信の高速化が実現した．

詳細な検証はこれからであるが、無線通信規格 IEEE802.11n を用いた呼び実験ではフレームレートが 2 倍程度向上した．

5. まとめ

ユビキタスに操作端末とのインタラクションを実現するため Web ベースインターフェースを用い，携帯端末のタッチインターフェースを使用することで携帯端末を用いた高精細タイルドディスプレイシステムを実装した．タイルドディスプレイシステム上では熱拡散シミュレーションを動作し大規模計算を計算性能の低い携帯端末上で高精細表示することに成功した．今後は目標最高解像度でのタイルドディスプレイの実装やクライアント端末のセンサ情報等を活用したディスプレイ領域の動的割り当て，クライアント端末の性能を考慮した領域サイズ決定，送信画像データの圧縮についても検討を行っていく予定である．

謝辞 本研究の一部は, JSPS 科研費 25280042 ならびに 25540043 の助成を得て実施しました．また, 数々の有力な御指導, 御意見を頂いた松山幸雄技術職員に心から感謝致します. 上記の皆様方をはじめ, 日頃様々な面でお世話になった本学森・福間研究室の皆様に深く感謝致します.

参考文献

[1] W.D. McCarty, S. Sheasby, P. Amburn, M.R. Stytz, C. Switzer, “A virtual cockpit for a distributed interactive simulation,”IEEE Computer Graphics and Applications, Vol. 14, Issue 1, pp.49–54, 1994.

[2] R. Marshall, J. Kempf, S. Dyer, ”Visualization Methods and Simulation Steering for a 3D Turbulence Model of Lake Erie,” Proc. Symp. on Interactive 3D Graphics, pp.89–97, 1990.

[3] C.Johnson, S.G. Parker, C. Hansen, G.L. Kindlmann, Y. Livnat, “Interactive Simulation and Visualization,” IEEE Computer, Vol. 32, No.12, pp.59-65, 1999.Dec. 1999.

[4] Q. Zhu, G. Agrawal, ”Supporting Fault-Tolerance for Time-Critical Events in Distributed Evironments”, Proc. of Supercomputing, Paper Nr. 32, pp.1-12, 2009.

[5] P. Malakar, V. Natarajan, S.S. Vddhiyar, ”An Adaptive Framework for Simulation and Online Remote Visualization of Critical Climate Applications in Resource-constrained Environment”, Proc. of Supercomputing, Paper Nr. 295, pp.1-10, 2010.

[6] P.R. Woodward, D.H. Porter, J. Greensky, A.J. Larson,M. Knox, J. Hanson, N. Ravindran, and T. Fuchs, “Interactive Volume Visualization of Fluid Flow Simulation Data,” PARA’06 Workshop on the State-of-the-Art in Scientific and Parallel Computing, 2006.

[7] 坂井陽平, 福間慎治, 浅野琢也, 森眞一郎:”高精細タイルドディスプレイを用いた並列ボリュームレンダリングシステムの実装”, 情報処理学会論文誌

[8] 荒川文貴, 福間慎治, 辺見良太, 森 眞一郎:”WebGL を用いた対話型遠隔シミュレーションシステムのマルチクライアント対応”, 先進的計算基盤システムシンポジウム SACSIS2013

[9] Byungil Jeong; Renambot, L.; Jagodic, R.; Singh, R.; Aguilera, J.; Johnson,A.; Leigh, J.; , ”High-Performance Dynamic Graphics Streaming for Scalable Adaptive Graphics Environment, ” SC 2006 Conference, Proceedings of the ACM/IEEE.