対話型遠隔シミュレーションのための Web ベースインタフェースの実装

山元祐弥[†] 辺見良太[†]
福間慎治[†] 森眞一郎[†]

1. はじめに

我々は, 遠隔地にあるサーバ上で実行中のシミュレーションに対して, クライアント端末から対話的な操作を可能にする対話型遠隔シミュレーションシステム[1)]の開発を行っている (図 1). 今までは, クライアント端末として研究室内のデスクトップコンピュータを想定したシステムを構築してきた. これに対し, 本研究はクライアント端末として, 近年著しい性能向上が見られるタブレット型コンピュータや, Android 端末等の携帯端末を用いることで, ユビキタスにリモートサーバとのインタラクションを可能にするインタフェースの携帯が目的である.

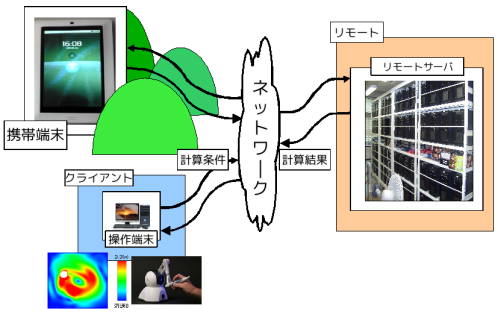

図 1 対話型遠隔シミュレーションシステム

2. 利用可能な技術の調査

2.1 要求定義と予備調査

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.

† 福井大学

表 1 調査結果

名称	動作内容	動作環境	要件
VRML	3D 描画処理	専用プラグイン	1,4
Canvas	2D 描画処理	HTML5 (JavaScript)	1,4
WebGL	3D 描画処理	OpenGL ES (JavaScript)	1,2,4
Ajax	サーバ間通信	XML (JavaScript)	1
Comet	擬似双方向通信	Ajax+ Long polling	1,3
WebSocket	双方向通信	WebSocket	1,3,4

(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（表 1 参照）. 調査の結果, 本研究では Canvas または WebGL[2)] と WebSocket[3)] 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.

2.2 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 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

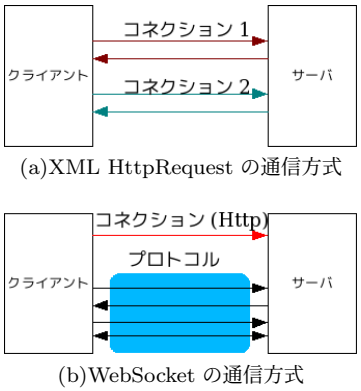

図 2 双方向通信機能の実現方式

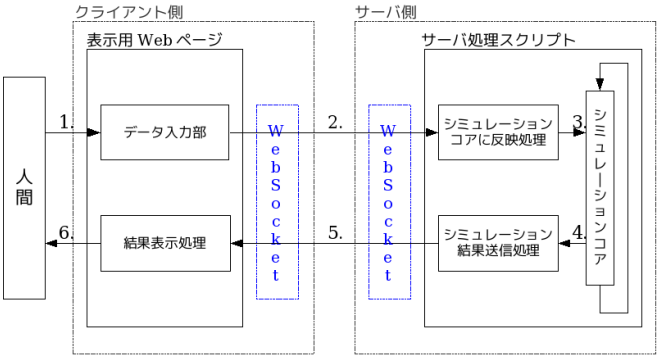

図 3 システムの概要図

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.

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.

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 (図 2) 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.

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（図 3 の 1,2) 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（図 3 の 4,5), Web ブラウザに表示されている画像が更新される（図 3 の 6)。

4. 実 装 結 果

Canvas と WebGL の 2 種類の Web ベースインタフェースを設計し, 様々な性能の PC を用いて評価実験行なった. クライアント側を Android 携帯端末としサーバ内部サイズ 64*64 の熱源活動を伴う熱拡散シミュレーションを行なった場合, 受信データを画面に反映する処理の部分に差が生まれ, 描画時間が Canvas 側が 51.08ms,WebGL 側が 1.18ms となった. 一方で,GPU の性能と比べて相対的に高性能な CPU を搭載した PC で実験を行なった場合には,Canvas の方が速いという結果も現れた.

5. ま と め

GPU が無い場合の Canvas,GPU がある場合の WebGL 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.

参 考 文 献

1) 橋本健介, 手塚俊作, 森眞一郎, 富田眞治:” シミュレーションキャッシングと遠隔インタラクティブ流体シミュレーションへの応用”, IPSJ Symposuim Series(SACSIS'09),Vol.2009,No.5,pp. 229–238.
2) WebGL Specification Ver.1.0, 10 Feb. 2011.
3) The WebSocket API Editors's Draft 8 Feb.2012, (http://dev.w3.org/html5/websockets/)