I-021

WebSocket を利用した並列レンダリング環境の構築
Construction of a Parallel Rendering System using WebSocket

南雲 佑介† 郡司 貴之† 長澤 可也†
Yusuke Nagumo Takayuki Gunji Kaya Nagasawa

1. はじめに

現存していない寺院などは実物を見ることが出来ず，全体をイメージすることが困難である．コンピュータグラフィックスの技術が活用されるまでは，模型などによる復元によって消失した寺院を再現していた．現在ではコンピュータグラフィックスによる復元が盛んに行われてきている．

コンピュータグラフィックスを用いるメリットとして，データがあれば復元した寺院をいくつでも作成出来ること，新しい情報があればデータの修正も容易に行えること，寺院の内側からのアングルなどの模型では見づらい視点などからも見ることが出来るなどが挙げられる．画像データや 3D データを Web により公開し，多数の人に閲覧してもらうことも出来るようになる．このようなメリットの反面，コンピュータグラフィックスでのアニメーションは，モデリングデータからレンダリングして作成される画像を連続で表示して作る．長時間のアニメーションを作るには使用する画像の枚数も膨大な数になり，レンダリングに多くの時間がかかってしまうという課題も残されている．

本研究では WebSocket によってコンピュータを複数台接続して並列レンダリングを行い，容易に 3D アニメーションを作成出来る環境の構築手法の提案をする．稼働実験では神奈川県鎌倉市二階堂の永福寺を対象にした．永福寺は 1185 年に源頼朝が建立を指示し，1192 年に本堂が完成した鎌倉の三大寺院にも数えられる寺院である．1405 年の火災によって焼失してしまっている．最近の調査で翼廊などが発見され，国の指定史跡となっている．これまで永福寺の復元において複数台を用いたレンダリングをする際は，モデリングデータを一旦サーバに保存し，1 台 1 台のコンピュータを操作して FTP でダウンロードを行っていた[1]．その後，すべてのコンピュータにアニメーションの何フレームから何フレームの間のレンダリングを行うかを 1 台 1 台別々に入力する必要があった．

2. レンダリングとは

レンダリングとは，3D オブジェクトを組み合わせてモデリングされたデータを 2D の画像として描画することを言う．レンダリングを行うソフトウェアをレンダラーと呼ぶ．レンダリングではモデリングされた物体の形や材質，それを捉える視点，光源の数や位置から生まれる影や光の反射など，画像を高画質にするには複雑な計算をしなくてはならない．3D アニメーションは，時間の経過による映像の変化をフレームで考え，ひとつのフレームにレンダリングされた 2D の画像を 1 枚表示する．次のフレームには時間の経過によって少し変化した画像を表示する．それを連続で行うことで 3D アニメーションとなる．図 1 のように右に進む車のアニメーションを作る場合，最初に左に位置する車，次に中央まで進んだ車，最後に右まで進んだ車，というように時間の経過によって車の位置が変化した画像を作る．それを順番に表示することで車が左から右に進んだというアニメーションになる．

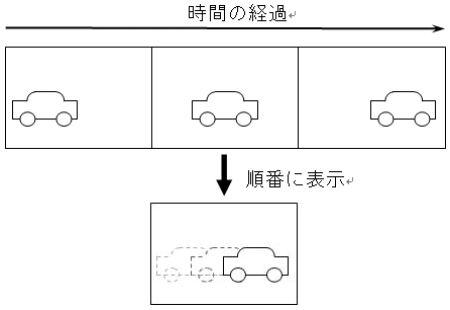

図 1: フレームの概念図

アニメーションはフレームの数によって滑らかさが変わる．フレーム数を多くすることでより細かな表示が行われ，アニメーションは滑らかになる．

このように長時間の 3D アニメーションを高画質で滑らかに作るには，高い性能のコンピュータと多くの時間が必要になる．

3. システム概要

本論文では複数台のコンピュータをマスタワーカ方式で接続しての並列計算を，WebSocket を用いて行う手法の提案をする．

WebSocket はサーバとウェブブラウザ間で双方向通信をするための通信規格である．2008 年に草案が公開された当初は HTML5 の一部として開発された技術であったが，現在は HTML5 とは別の規格として仕様策定が進められている．Safari や Google Chrome などの先進的ないくつかのブラウザでは WebSocket の利用が可能であり，iPhone や iPad でも iOS4.2 以降で WebSocket が利用できる．ブラウザ上で双方向通信を行うための従来の方法であった Ajax では，ワーカ側がマスタ側へ頻繁に受信する情報があるかを確認する必要があり，それにはネットワークやコンピュータのリソースを多く消費してしまう．Websocket では接続を行うまでを HTTP によって行い，接続が行われてからは負荷の少ない専用のプロトコルを利用しサーバとクライアント間の双方向通信をリアルタイムで行うことが出来るようになる．並列計算を行う際に問題となる環境の違いなどを考慮せずに済むようにし，容易に並列計算を行えるようになる．また，WebSocket の低負荷でのリアルタイム通信を利用して各ワーカの進捗情報を即座に知ることも可能となる．WebSocket については[2]に詳しく載っている．

マスタからの指示は JavaScript で行われ，ワーカは PHP によって受け取る．マスタは Safari や Google Chrome などの WebSocket に対応したブラウザならば動かすことが出来るので，iPad や Android 携帯も利用することが出来る．

†湘南工科大学, Shonan Institute of Technology

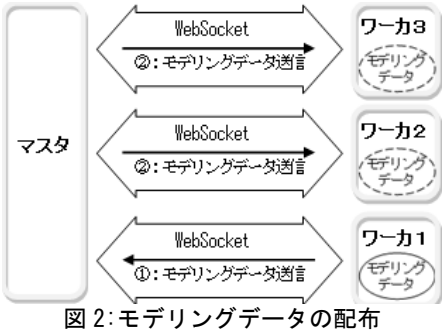

図 2:モデリングデータの配布

図 2 と図 3 は提案するシステムの概要である．マスタの役割をするコンピュータ 1 台とワーカの役割をするコンピュータ複数台との通信が WebSocket で繋がれている．すべてのワーカには Maya 及び Maxwellrender などのレンダラーがインストールしてある．ワーカ 1 にレンダリング元のモデリングデータが入っていて，図 2 の①でモデリングデータをワーカからマスタへ送信する．受信したマスタは図 2 の②のようにワーカ 2 とワーカ 3 にデータを配布する．

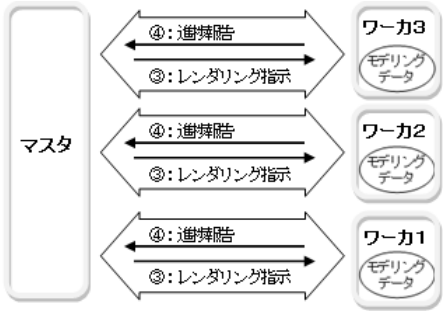

図 3:レンダリングの指示

図 3 の③のように，マスタはフレームを指定してレンダリングをするようにワーカへ指示を出す．指示を受け取ったワーカはレンダリングを実行する．図 3 の④のようにレンダリングが何フレームまで終わったかの進捗状況をマスタへ報告し，終了したワーカはマスタから次にレンダリングを行うフレームの指示を受けレンダリングを行う．

4．稼働実験

今回の稼働実験をマスタ 1 台，ワーカ 2 台で行った．各ワーカにはレンダリングソフトウェアの Maxwellrender が入っていて，3D アニメーションソフトウェアの Autodesk Maya で作ったモデリングデータがワーカの 1 つに入っている．本システムを使いモデリングデータをすべてのワーカへ入るようマスタからの指示で送信させ，フレームを指定して並列にレンダリングを実行させた．

図 4:マスタ画面

図 4 は実験時のマスタの実行画面である．ブラウザは Safari を使いモデリングデータのあるワーカの IP アドレスとレンダリングしたいワーカの IP アドレスを指定する．前者は図 2 の一番下のワーカを意味する．入力欄の下にあるボタンを操作してモデリングデータの圧縮・配布・展開，レンダリングの実行を各ワーカへ指示する事が出来る．レンダリングの結果として図 5 のような画像が得られた．

図 5:レンダリング画像

稼働実験ではモデリングデータの指定などは直接マスタ側のプログラムに書いたが，JavaScript に入力欄と動作を追加すればモデリングデータの入っている位置を指定することができる．

5．考察

今回は稼働実験で WebSocket を使った並列レンダリング環境の構築が出来た．マスタに JavaScript で記述されたプログラムを，ワーカに PHP で記述されたプログラムを入れておけば，後の操作はマスタからの操作だけで済むことが確認された．これにより，今までの作業と比べてより容易に並列レンダリングを行えるようになった．

6．まとめ

今回の稼働実験では少ない台数での実験だったので，より多数のワーカを繋いでの実験を行う必要がある．また，このシステムはレンダリング以外でも使えるはずである．並列に行うことでメリットが生まれる他の場面での実験も行いたい．

参考文献

[1] 長谷川聡，杉本直也，成岡浩二，長澤可也，“歴史的文化財の 3D 技術による復元”，情報処理学会第 72 回全国大会論文集，第 4 分冊，pp.397-398，2010.
[2] 小松健作，“徹底解説 HTML5 API ガイドブック コミュニケーション系 API 編”，秀和システム，2010.