K-071

対話型動的 3DCG システム
Interactive Dynamic 3DCG System

柳瀬龍郎† 松田大樹† 田村信介†
Tatsuro Yanase Daiki Matsuda Sinsuke Tamura

１．まえがき

計算機シミュレーションが行われる場合，通常その目的から幾つかの実験条件を変えてシミュレーションが繰り返される．実験条件の変更は，いわゆるパラメータ，すなわち複数の変数あるいは係数の値を変更する事により行われる．シミュレーションは，(1)前処理におけるパラメータの設定または変更，そして(2)実験によるデータの取得，そのあと(3)後処理でグラフィックなどによる出力と分析・解析が行われる．更には，必要な出力が得られるまで，このサイクルの反復が行われる．このサイクルのうち (2) のステップに費やされる時間は計算機システムの性能により，決定される要素が大きい．このための手段として MPI 等のプログラミングライブラリが近年次第に整備されつつある[1]．シミュレーションの目的は，モデルを支配する重要因子の選択・決定である．従ってどのパラメータが実験結果にどのように影響し結びつくのか，シミュレーション入力方法と出力表現が直感的に把握しやすいことが，求められる．この意味において(1)(3)のステップは実験におけるヒューマンインタフェースとしての極めて重要な部分である．

上記の機能実現を目的とし，本研究では近年のパーソナルコンピュータの高性能なグラフィック表示機能と，マウスなどのポインティングデバイスを用いる事により，これまでは高価な計算機システムでしか実現しえなかった，対話型動的 3DCG 表示システムの提案と実装を行う．

2. クライアントサーバシステム

仮に計算機が高速に数値実験を処理し，この結果“対話的”にシミュレーション実行が可能になれば，この(1)(2)のステップは理想的なものとなる．(1)で利用できるデバイスとしてはマウスやジョイスティックあるいはデータグローブ等が利用可能である．(3)のシミュレーション結果は，近年視覚的に判断しやすい CG でなされる事が多くなってきている．CG 表現には静的表現と動的表現があるが，特にシミュレーションの結果が時間の関数で表される場合には出力表現がディスプレイ上のアニメーション，すなわち動画が用いられる．この場合，シミュレーションモデルのパラメータが修正されたなら，当然の結果としてシミュレーション出力表現としての動画も異なったものとなる．このパラメータ修正入力に対応した，シミュレーションの結果としての動画の変化は，パラメータのモデルの構成因子としての影響力等を，数値シミュレーションの結果としての膨大な量の数値を読み取るより，はるかに容易に把握しうる利点がある．

このような考えに基づき，本研究では入力と動的 3DCG 出力を，ヒューマンインタフェースを考慮した高機能な一つのシステムとして構築することとする．この機能実現のために，システム全体を数値シミュレーション処理用計算サーバ（以下計算サーバ）と，3D グラフィック表示が可能なクライアント（以下表示システム）に分けたサーバ・クライアントモデルとする．

†福井大学 工学部 情報・メディア工学科

３．表示システムの道具立てと構成

3D 表示はその目的上，奥行きや凹凸感がリアルに認識される必要がある．このためにディスプレイ装置そのものについても，２次元表面以外の方法が求められ，研究されている．いずれの場合も，容易に視点を変えて対象の立体的視覚イメージを認識できることが 3D 表示装置の本質的な要件である．本研究では可搬性を重視し，すべてソフトウェアによって処理が可能である等の理由により，２次元ディスプレイ上での動的 3DCG 表示とする．

従来，計算結果を 3D 表示する場合，計算機メーカーなどによって提供される専用ライブラリを利用するか，あるいは OpenGL や DirectX などを用いて低レベルから記述されるのが一般的であった．しかし，動的３ DCG 表現には記述コストの観点からこれらの方法は不利であり高機能な記述言語を用いる事が求められる．

３次元バーチャルリアリティのシーン表示にはいくつかの高級記述言語が存在し，筆者らはこれまでにも VRML 等を用いた研究を行ってきた[2][3][4][5]．しかし，例えば VRML では基本的に 3D オブジェクトの動的変形についての機能が脆弱である．より柔軟なシステムの実現を目指す本研究では，Java とその 3D 表示のための標準拡張 API である Java3DAPI を採用する．Java3DAPI は高レベルなシーンの記述が可能である．例えば，視点の移動，物体の移動，物体の色/透明度の変化そして物体の変形等のための基本的機能が装備されている．

一方入力については，3D 表示されたバーチャル空間と操作者，あるいは現実世界とのインタラクションは，出力結果を評価/判断し，条件を変えて再度シミュレーションを反復する．本提案システムでは，マウスなどのデバイスを使い GUI(Graphical User Interface)を通して対話的にパラメータを修正/設定し，さらに表示されたバーチャル空

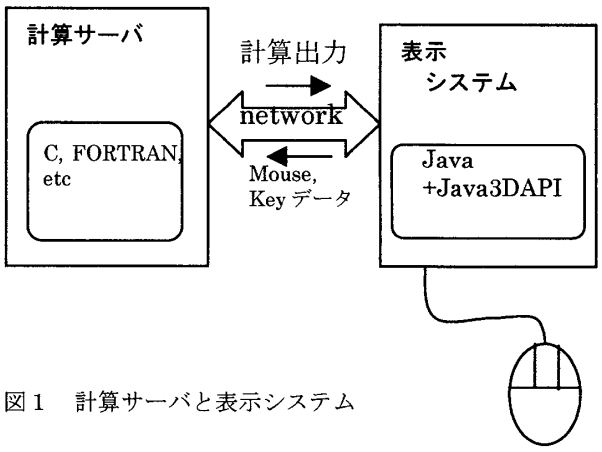

図１　計算サーバと表示システム

間における出力の視点の移動も，マウスにより実現する．計算サーバでは計算を行い，その結果のデータを表示システムに送信する．表示システムはキーボードやマウスからの入力を計算サーバに伝えるために，データを送信する．交換されるデータの「型」はプリミティブな４バイトの整数に限定する．これは，既存のシミュレーションのプログラムがC言語などで書かれている場合でも，本システムを利用する場合において，データ交換を容易に実現するためである．

表示システムは交換されたプリミティブなデータを解釈・実行し 3D 空間の表示を行うが，以下のような機能を実装する：

- シミュレーション結果をネットワーク経由で獲得
- シミュレーション結果を随時 3DCG で表示
- パラメータなどをリアルタイムかつ対話的に変更
- 3DCG 構成（視点，色，透明度など）は対話的に変更
- 対話処理は GUI（ボタン，スライダー，マウスジェスチャー等）による
- 必要に応じて表示システム側で物体や光源を 3D 空間に追加

表示システム全体は３つの部分で構成される．

(1) ネットワークの処理部はリモートの計算サーバと通信し，データを得る．

(2) ユーザ入力監視部で受信され，リモートに伝達する必要があるもの（シミュレーションのパラメータ変更など）はネットワークの処理部によりデータ交換される．ユーザ入力は GUI による簡単な操作で可能である．

(3) 表示は 3D 空間更新部によって随時更新される．ユーザ入力のうち，リモートに伝達せずローカルに処理するもの（視点，物体の透明度変更など）は更新部で実行される．

通信や 3D 空間更新はマルチスレッド化し，効率よく処理を行う．

視点の変更や物体の透明度の変更機能は，表示システム側で基本的に装備可能である．従って，シミュレータのアプリケーションプログラマが 3DCG 表示コードを記述する負担軽減になる．さらには，表示システム PC のローカル処理により行われるために，アプリケーションのシミュレーション計算負荷にも影響が及ばない事等が本システムの最大の特徴であり利点である．

４．表示支援ライブラリの機能と構成

システムの計算サーバ側では，計算処理の他はデータの送受信を行うのみであり，既存のライブラリを呼び出す程度で機能を実現できる．一方の表示システム側では計算サーバとのデータの交換の他に，受信したデータを元に 3D 空間を構成し，表示する機能が求められる．また，外部(あるいは操作者)とのインタラクションを行い，これに基づいて適切に反応を返さなければならない．この機能の実現のために，Java3DAPI を用いる．また，3D 空間の構築を容易にするためにライブラリを作成した．このライブラリの機能は：

1) ネットワークを通じて計算サーバとのデータの交換
2) 動的 3D 空間構築のための柔軟な支援機能提供

である．また，3D 空間構築を支援するためのライブラリは，主に：

(1) 3D 空間表示のためのデータを生成すると宣言されたインタフェース群
(2) (1)のシンプルな実装クラスの集まり
(3) (1)のインタフェースを解釈・実行・自動更新する javax.media.j3d.Node を作る StageBuilder クラス
(4) StageBuilder クラスで作られた javax.media.j3d.Node を含む 3D 空間を“シンプルに”表示するための GUI を生成する SimpleMainFrame クラス

から構成される．

5 ライブラリの利用

本研究で提供するライブラリの一般的な利用手順は；

1) Factory インタフェースを間接的に実装
2) 1)で実装されたクラスのインスタンスを生成
3) 2)で生成されたインスタンスを StageBuilder クラスのコンストラクタに渡し StageBuilder のインスタンスを生成
4) GUI 生成をライブラリで行うには 3)のインスタンスを SimpleMainFrame クラスのコンストラクタに渡しインスタンスを生成．StageBuilder クラスは javax.media.j3d.Node を生成する．
5) SimpleMainFrame クラスには GUI 画面を可視化する showFrame()メソッドが用意されているのでこの実行により GUI 画面を表示．

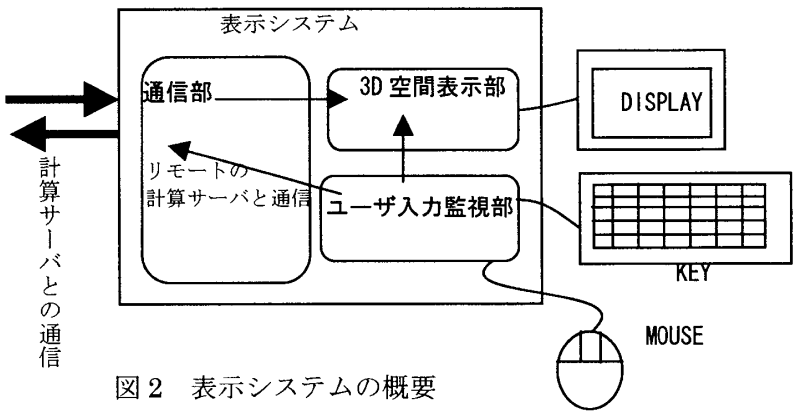

図２　表示システムの概要

リスト１にライブラリの利用例を示す．このコードでは三つの座標のみを与えている．

まず，TestFactory クラスでは FloatArrayFactory インタフェースを実装する．Factry インタフェースとは，このライブラリにおいて何がしかのデータを生成すると宣言されているインタフェースである．

FloatArrayFactory は float(4 バイト実数)配列のデータを生成する Factory のサブインタフェースである．TestMain クラスは main メソッドを持つメインクラスである．リスト１の出力を図 3 に示す．図では 3 点がそれぞれ矢印の方向に移動する．

６．サンプルプログラム

別のサンプルプログラムの実行結果を図 4 に示す．この例では，10×10 の質点を持つ 2 次元バネモデルの一点支持自由運動の動的シミュレーションを行っている．表示クライアントにおける出力表示が図 4 である．フレーム中央にはこのシミュレーションによって得られた格子模様の物体があり，その奥に立方体が見えている．この立方体は動的シミュレーションとは関係ないが，分かりやすくするために表示システム側で追加している．フレーム上部のスライダーで格子模様の物体の透明度を変更できる（左に行くほど透明になる）．この GUI のスライダーによる透明度変更指示や，マウスジェスチャーと呼ばれる操作による視点の変更指示はリモートの計算サーバには伝達されず，ローカルな表示システムで処理され，直ちに描画に反映される．また，フレーム下部のスライダーはこのバネモデルのバネ定数値を変更できる（左に行くほど弱いバネになる）．このスライダーの動きは数値として計算サーバに伝えられ，モデルのバネ定数値がリアルタイムに変更され，計算シミュレーションに影響する．また，ここで用いたサンプルプログラムではキーボードで，格子模様の物体の端をつかんで引っ張ったりできる（図右下）．このキー入力情報も計算サーバに伝えられ，直ちにシミュレーション処理に影響を与え，そして表示に反映される．スライダーはアプリケーションに複数個配置可能であり数に制限はない．また，サンプルプログラムにおけるバネ定数のように，スライダーにより制御される値の制限範囲も自由に設定可能である．また，スライダーによって与えられた数値を表示する事も，またスライダーによらず直接数値を与える事も当然可能である．

これらのパラメータの設定・修正を行う GUI とシミュレーション結果の 3DCG の表示が同一表示画面上に存在する事が大切であり，さらには，視点の移動が計算サーバの負荷を増大させる事なくローカルに処理され結果表示される事がいわゆる“使い勝手の良いシミュレータシステム”である要件を満たしていると考えられる．

サンプルプログラムの実行環境は：

表示システム側 PC：

OS:Win2k, CPU:セレロン 2.4GHz, mem:256Mbyts

計算サーバ側 PC：

OS:Linux, CPU:Athron1.6GHz×2, mem:512Mbyts

ネットワークは 100MHz Ethernet.

```
//本研究のライブラリのインポート
import jp.ac.fukui_u.fuis.radio.matsuda.j3dnet.*;
//Adapter を継承し、Factory を間接的に実装するクラス
//Adapter は Factory インタフェースのアダプタクラス
class  TestFactory  extends  Adapter
            implements  FloatArrayFactory{
//FloatArrayFactory から得られる値 float[]の要素は
//3n は n 番目の物体の x 座標、3n+1 は y 座標、3n+2 は z 座標
private float value = new Float[]{ 0.0f, 0.0f, 0.0f, // 0 番目の座標
                                   2.0f,2.0f,3.0f, // 1 番目の座標
                                  -3.0f,1.0f,-5.0f};// 2 番目の座標
//FloatArrayFactory から継承したメソッド
public float[] getFloatArray() {return value;}
//Factory と Adapter から継承したメソッド
// (値を少しずつ変化させる)
public void update() {
  f[0] += 0.001f;  //0 番目の物体の x 座標値が変化（画面右に動く）
  f[4] += 0.001f;  //1 番目の物体の y 座標値が変化（画面上に動く）
  f[8]－= 0.001f;  //2 番目の物体の z 座標値が変化（画面奥に動く）
  }
}

//メインクラス
 public class TestMain {
   public static void main(String[] args) {
     //Factory を生成
     FloatArrayFactory factory =  new TestFactory();
     //StageBuilder を生成
     StageBuilder sb = new StageBuilder(factory);
     //SimpleMainFrame を生成、可視にする
     SimpleMainFrame frame = new SimpleMainFrame(sb);
     frame.showFrame();
   }
 }
```

リスト１　ライブラリの利用例

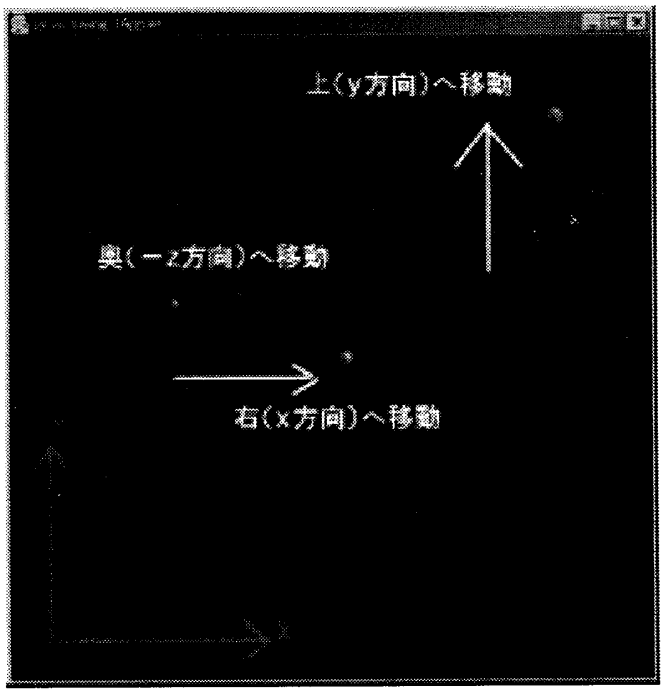

図３　リスト１の出力

ライブラリは全体で約 2500 行の Java で記述され，また約 550Kbyts の Help 情報を備えている．

７．まとめ

ネットワーク環境用においても使用可能な対話型動的 3DCG システムを提案し実装した．本研究において Java と Java3DAPI を用い，動的 3DCG 表示を支援するためのライブラリを新たに構築用意した．また，本報告においてこのライブラリの使用例と，ライブラリを使用したサンプルプログラムにより，対話的でかつ動的シミュレーションを行い，それが滑らかに表示される事を示し，システムの有効性を確認した．

本研究の提案システムを用いたシミュレータアプリケーションを構築する事により，シミュレーションが実行開始されると，クライアントとサーバの間で情報交換しながら，全体として 3DCG シーン/物体の座標を動的に計算し表示させ，また GUI による対話的視点の変更，パラメータの修正・設定が可能となり，全体として計算シミュレーションの実行そして動画表示をシームレスに進めることができる．

今後は，より大規模な動的 3DCG シミュレーションに本研究の成果を利用しその有効性を拡張したい．

８．参考文献

[1]Message PassingInterface Forum,” MPI:A message-passing interface standard”, Int. J. Supercomputer Applications and High Performance Computing, vol.8, no.3, pp.159-416, 1994.

[2]清川貴氏“異種言語による VRML ワールドの操作”，平成 13 年度福井大学工学部卒業研究(2002.3)

[3]中村真人”バネモデルによる VRML の変形シミュレーション”，平成 14 年度福井大学工学部情報・メディア工学科，卒業研究(2003.3)

[4]松田大樹，田村信介，柳瀬龍郎 “ネットワーク越しで使える 3 次元表示システム”，平成 15 年度電気関係学会北陸支部連合大会，F-41(2003.9)

[5]柳瀬龍郎，松田大樹，田村信介“並列・分散処理用 3D 表示関数とクラス”，電子情報通信学会信学技報，MVE 2003-104，119-124 (2003.11)

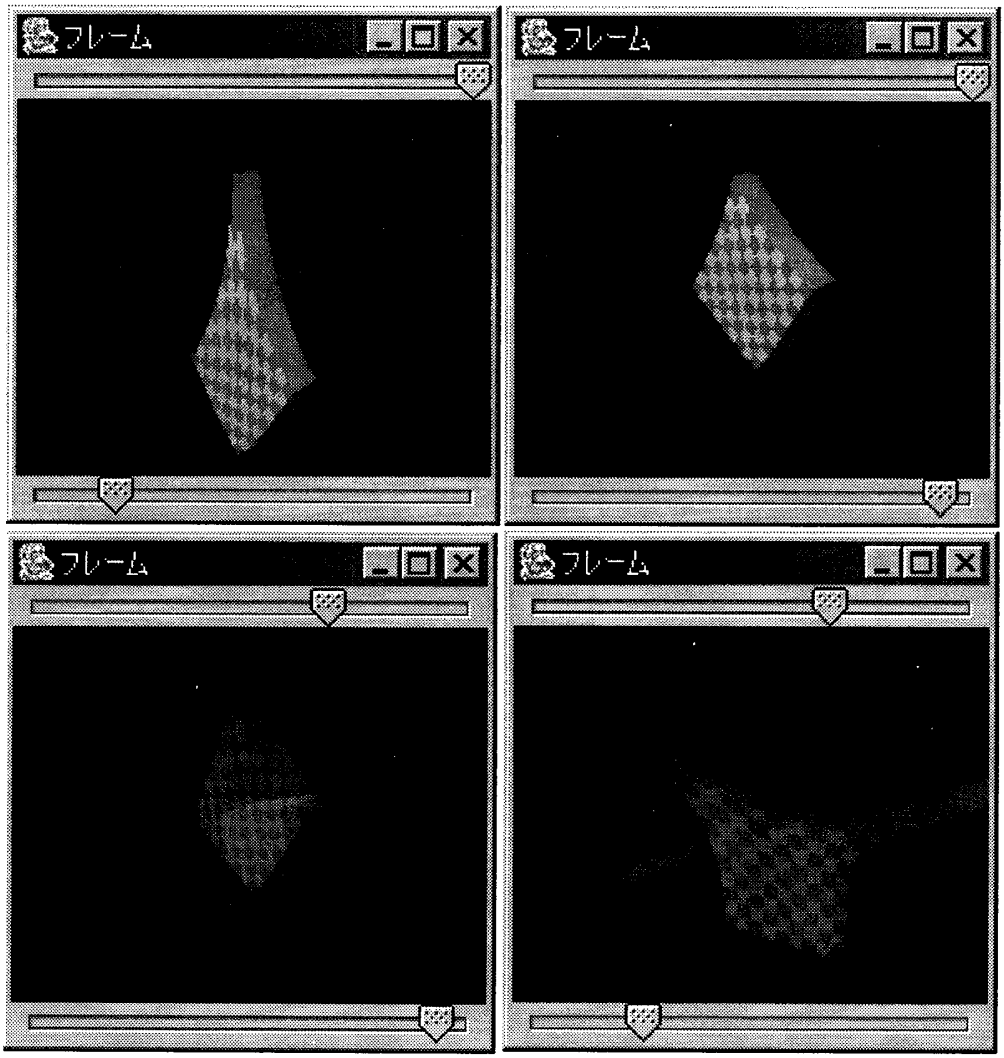

図４　サンプルシミュレーションプログラムの出力