第 18 回日本バーチャルリアリティ学会大会論文集(2013 年 9 月)

モバイル MR システム構築のための 機能分散型フレームワーク(8) －アニメーションの表現力向上－

A Distributed Framework for Mobile Mixed Reality System (8) - Design and Implementation of Animation Mechanism -

川端 大輔，木村 朝子，柴田 史久
Daisuke Kawabata, Asako Kimura, and Fumihisa Shibata

立命館大学大学院 情報理工学研究科（〒525-8577 滋賀県草津市野路東 1-1-1）

概要: 我々の研究グループでは，クライアント・サーバモデルを用いて複数の端末間で MR 空間を共有可能なモバイル複合現実感システム構築のためのフレームワークを提案している．しかし，提案フレームワークでは，変形しない CG モデルをスクリプト言語の記述により制御することでアニメーションを表現するため，オブジェクトの形状が変化するアニメーション表現は記述が煩雑になるといった課題が存在した．そこで本稿では，提案フレームワークにおいて，ボーンアニメーションおよびパーティクルを用いた表現手法を設計・実装した結果について報告する．

キーワード: 複合現実感，モバイル，フレームワーク，アニメーション

1. はじめに

近年，モバイル端末を利用した複合現実感（Mixed Reality; MR）への期待が高まりつつある[1]. そこで，我々はこれまで，複数の端末が同一の MR 空間を共有可能なモバイル MR システムのためのフレームワークを設計・開発してきた[1]. 本フレームワークは，クライアント・サーバモデルを採用し，MR 空間中に存在する 3DCG などの仮想オブジェクトの情報をサーバが一元管理することで複数のクライアントで同一の MR 空間の共有を可能とする. 仮想オブジェクトの動作は，独自に設計したスクリプト言語により，複雑な動きの表現や物理法則に従った動作を容易に記述可能としている[3]. しかし，仮想オブジェクトのアニメーションは全て変形しない 3DCG モデルによって表現するため，オブジェクトの形状が変化するアニメーション表現は記述が煩雑になるといった課題が存在した.

そこで本研究では，ボーンアニメーション及びパーティクルを用いた表現手法を導入し，フレームワークの表現力向上を目指す．具体的には，3D アニメーション作成ソフトウェアで作成したモーションデータを読み込み，スクリプトによってそれを制御することでアニメーション表現のスクリプト記述量を削減する．また，パーティクルを用いた表現の型を用意し，スクリプトでパラメータを記述することでパーティクル表現を可能とする．

2. コンテンツ制御機構

2.1 制御手法

本フレームワークのコンテンツ制御機構は，サーバに集約されており，スクリプトを逐次実行することで MR 情報を更新する．クライアントは，サーバから MR 情報を定期通信により取得し，クライアント自身の位置・姿勢を基に，画像をレンダリングすることで MR 提示を行う．

2.2 コンテンツの構成

本フレームワークで扱うコンテンツとして，クライアントとオブジェクトが存在する．オブジェクトは仮想オブジェクトと実オブジェクトに大別される．仮想オブジェクトは，1 つ以上のコンポーネントの集合として構成される．コンポーネントは，3D モデル・画像・テキスト・サウンドを扱うことができる[4]. これらのコンポーネントが**図 1** に示すような階層構造をとることで，仮想オブジェクトを構成する．

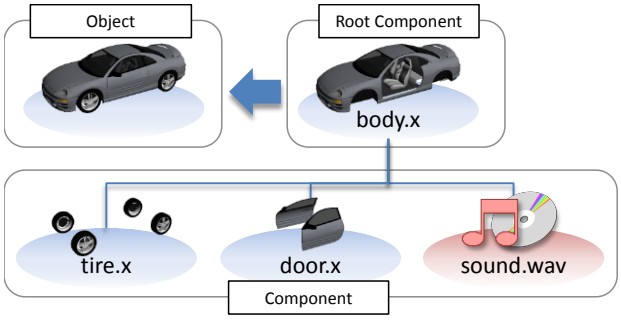


図 1 仮想オブジェクトの構成図

2.3 アニメーション表現における問題点

本フレームワークの仮想オブジェクトは,スクリプトに位置・姿勢などのパラメータの変更量を記述しておくことでアニメーションを表現することが可能である．しかし，仮想オブジェクトの 3D モデルのアニメーションは，変形しない CG モデルを用いて表現するため，オブジェクトの各部位を個別に動作させたい場合，部位をコンポーネントとして分割し，個々のコンポーネント毎にパラメータの変更量を記述する必要がある．そのため，動物の手や足など，可動部位の増加に伴い記述が煩雑になるといった問題が存在する．また，煙や炎など，形状が不規則なオブジェクトを表現しようとすると，多数のコンポーネントが必要であり，処理が複雑となるといった問題も抱えている．

3. アニメーション制御機構

3.1 設計方針

アニメーション制御機構では，上記で述べた問題点を踏まえ，従来では実現が困難であった表現を，スクリプトの少ない記述量により実現することを目標とする．そこで本研究では，以下の 2 つの制御方法を導入する．

- ボーンアニメーションを用いた表現
- パーティクルを用いた表現

これらの表現を，従来の制御手法と同様にスクリプトによって容易に制御可能とし，従来との高い互換性を実現するために，以下の設計方針を定めた．

(1) 従来の仮想オブジェクトの構成要素であるコンポーネントとしてボーンアニメーション及びパーティクル表現を追加する．

(2) アニメーションを行う上でのパラメータを事前に設定し，できるだけ簡単なパラメータ設定で，アニメーション表現を利用可能とする．

3.2 ボーンアニメーションの制御

本フレームワークで扱うボーンアニメーションは，既存のソフトウェアで作成されたモデルとモーションのデータを用い，スクリプトにより制御する．これにより，従来のスクリプトでの煩雑なアニメーション記述を削減することが可能である．

3.2.1 ボーンアニメーションコンポーネント

ボーンアニメーションのモーションデータを扱うコンポーネントを新たに追加する．ただし，このコンポーネントは 3D モデルを扱うコンポーネントが子として持つものとする．

ボーンアニメーションを制御するために必要なパラメータを考慮し，スクリプトの設計を行った．まず，ボーンアニメーションを扱う上で最低限必要な操作として「再生」と「一時停止」「停止」が考えられる．次に，連続して再生するために「繰り返し再生」が挙げられる．また，モーションの速度を制御するために「再生速度」が必要である．これらの制御を可能とするために用意した変数を**表 1**にまとめる．また，ボーンアニメーションを扱うスクリプトの記述例を**図 2**に示す．

表 1 モーションを扱うコンポーネントのパラメータ

型	変数名	用途
int	*state*	再生(0)・一時停止(1)・停止(2)
int	*loop*	繰り返し再生回数
double	*speed*	再生速度

```
/* コンストラクタ */
public Object1(Position3D pos, Orientation ori){
  /* 初期設定 */
}
/* 再生するメソッド */
public void start (){
 this.root.motion1.state = 1;
}
/* 無限ループさせるメソッド */
public void loop (){
 this.root.motion1.loop = 0;
}
```

図 2 ボーンアニメーション制御のスクリプト記述例

3.2.2 ボーンアニメーション再生処理

(1) 再生処理

クライアントはボーンアニメーションを扱うコンポーネントの情報を受け取ると,3DCG モデルに対しボーンアニメーションを適応する．描画の更新の際に，ボーンアニメーションのフレーム番号を更新することでアニメーションを表現する．前述のパラメータの *state* が 1 であれば再生し，前回の描画から経過した時間 *time* [ms] を計測し，前フレームから更新するフレーム数 *frame* を式 (1) を用いて計算する．準備するボーンアニメーションは 30fps で動くものとし，更新するフレーム数に *speed* を掛け合わせて再生速度を反映する．これにより，処理速度の異なる端末間でアニメーションの再生速度のずれが起こることを防ぐことができる．

state が 2 であれば一時停止のため *frame* を 0 とし，フレーム番号の更新を行わない．*state* が 3 であれば停止処理としてフレーム番号を 0 とする．

$$frame = speed \times \frac{time}{1000} \times 30 \qquad (1)$$

(2) ループ処理

再生処理でフレーム番号を更新し，モーションデータの最終データを超えた場合，フレーム番号を戻し，*loop* 回数分再生を続ける．

3.3 パーティクルの制御

3.3.1 パーティクルコンポーネント

本フレームワークでパーティクルを汎用的に使用可能にするためにどのような表現が必要か検討する．今回は，パーティクルを用いた表現として煙と炎の表現を導入する．まずパーティクル表現には，「発生するパーティクル数」と「色」「初速ベクトル」「寿命」が必要である．更に，「時間による減衰」と，「重力場のベクトル」を設定することで，パーティクル数の減少と発生したパーティクルが場の影響を受ける表現を可能とする．これらの制御を可能とするために用意した変数を**表 2**にまとめる．また，パーティクルを扱うスクリプトの記述例を**図 3**に示す．

表 2 パーティクルを扱うコンポーネントのパラメータ

型	変数名	用途
int	*particleNum*	発生するパーティクルの数
double[3]	*color*	色
double[3]	*velocity*	初速ベクトル
int	*endTime*	寿命
int	*attenuation*	時間による減衰
double[3]	*gravity*	重力場のベクトル

```
/* コンストラクタ */
public Object2(Position3D pos, Orientation ori){
  /* 初期設定 */
}
/* 煙を生成するメソッド */
public void start (){
 Color color = new Color(50,50,50);
 Vector3d vec = new Vector3d(0.0, 10.0, 0.0);
 this.root.smoke.particleNum = 500;
 this.root.smoke.color = color;
 this.root.smoke.velocity = vec;
 this.root.smoke.endTime = 1000;
}
```

図 3 パーティクル制御のスクリプト記述例

3.3.2 パーティクル表現の描画処理

(1) パーティクルを生成

particleNum で指定した数のパーティクルを生成する．生成の際には各パーティクルに *color*，*velocity*，*endTime*，*gravity* と初期位置 *pos* にそれぞれ乱数を加えて設定する．

(2) パーティクルを更新

パーティクルが生成されてからの経過時間 *t* を用いて現在の位置 *pos'* を式 (2) を用いて計算する．また，*t* が *endTime* を超えた場合には，*attenuation* により消滅するか再生成するかを決定する．

$$pos' = pos + velocity \times t + \frac{1}{2} \times gravity \times t^2 \qquad (2)$$

(3) パーティクルの描画

各パーティクルにはパーティクルの位置を中心とする正方形の面ポリゴンを生成し，あらかじめ用意しておいたテクスチャを貼り付ける．その面ポリゴンをクライアントの位置姿勢により，画面に正対するように配置し，各パーティクルを描画する．

4. 同期処理

本フレームワークでは，サーバから通信により情報を受け取ることで，各クライアントでサウンドやボーンアニメーションを用いた表現が可能となった．しかし，サーバと各クライアント間の通信にかかる時間が異なるため，クライアント毎にサーバから情報を受信する時刻に差が発生する．この時刻の差によりサウンドやボーンアニメーションの再生開始タイミングが異なり，クライアント間にずれが発生してしまうという問題が存在する．そこで，サーバと全てのクライアントで時刻同期を行い，サーバからの MR 情報に時刻情報を付随してクライアントへと送信し，時刻情報を用いて補間処理及び再生タイミングを同期する機構を追加する．

4.1 時刻同期手法

クライアントは起動時に，サーバへと時刻同期のリクエストを送信する．サーバは，クライアントからのリクエストを受信した時刻 *Tsr* と，クライアントへとレスポンスを返す時刻 *Tss* をレスポンスとして送信する．サーバからのレスポンスを受信したクライアントは，リクエストを送信した時刻 *Tcs* とリクエストを受信した時刻 *Tcr* を用いて，サーバに対する時刻のずれ *diff* を，式 (3) を用いて計算する．ただし，通信の往路にかかった時間と復路にかかった時間が同じであると仮定する．

$$diff = \frac{Tcs + Tcr}{2} - \frac{Tsr + Tss}{2} \qquad (3)$$

4.2 補間同期処理

クライアントは，サーバから定期的に受信した情報を補間しながら描画する[3]．通信にかかる時間の差により発生するクライアント間のずれをなくすため，クライアントがサーバに対して遅延する時間を設定し，全クライアントで統一する．クライアントは，現在の時刻から設定した遅延時間分遡った MR 空間を，補間処理を用いて描画する（**図 4**参照）．

4.3 再生時刻同期処理

クライアントでサウンドやボーンアニメーションの再生時刻を統一するために，サーバからクライアントへと送信する情報に，再生開始時刻を付随して送信する．クライアントは，サーバから受信した再生開始時刻になれば再生を開始する．情報を受信した時刻が再生開始時刻を過ぎていれば，現在の時刻との差分を計算し，途中から再生を行う．この仕組みにより，通信にかかる時間の差に関係なく，全てのクライアントで同一時刻に再生することが可能となる．ただし，前述のサーバに対するクライアントの遅延時間を考慮する．

5. 動作確認

以上の設計によりアニメーション機構を拡張し，提案機構が正しく動作するか確認する．

【実験 1】ボーンアニメーションの動作確認

ボーンアニメーションを用いた表現が正しく動作するか確認する．実行したスクリプトを**図 5**に示し，実験結果を**図 6**に示す．

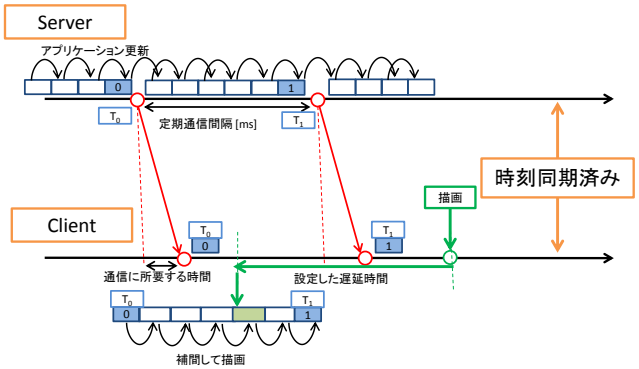
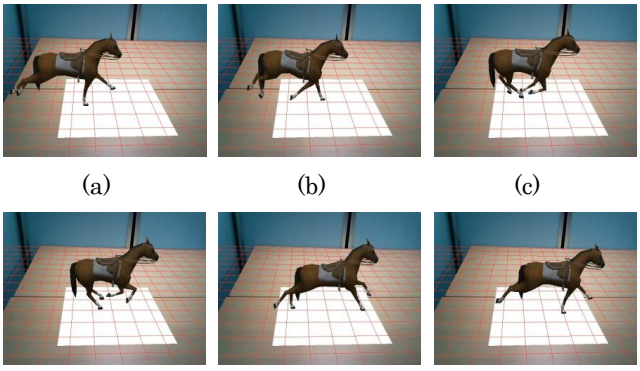


図 4 同期処理

```
/* コンストラクタ */
public Object1(Position3D pos, Orientation ori){
  /* 初期設定 */
}
/* 実行メソッド */
public void start (){
 this.root.motion1.state = 1;  // 再生
 this.root.motion1.loop = 0;   // 繰り返し回数無限
 this.root.motion1.speed = 1;  // 再生速度1.0倍
}
```

図 5 実験 1 で用いたスクリプト



(a) (b) (c)

(d) (e) (f)

図 6 実験 1 の結果

図 6 における(a)-(f)は時系列順である．結果より，正しくアニメーションが再生できていることを確認した．

【実験 2】パーティクル表現の動作確認

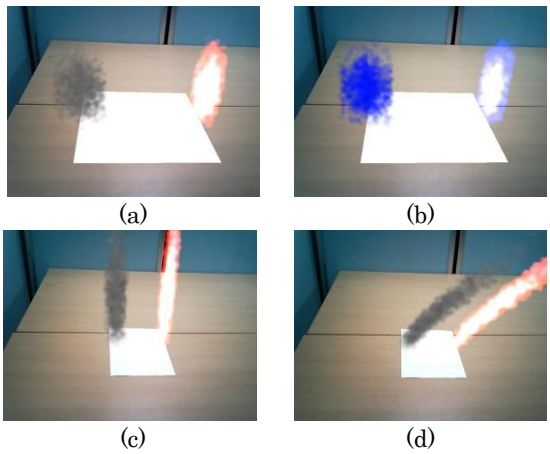
パーティクルを用いた表現が正しく動作するか確認する．パーティクルのパラメータとして，色，寿命，重力ベクトルを変更し，正しい表現が行われているか確認する．実行したスクリプトを図 7に示し，実験結果は図 8に示す．(a)はデフォルトのパラメータでパーティクルを生成した結果，(b)はパーティクルの色を変更した結果，(c)は寿命を 4 倍に伸ばした場合の結果，(d)は横向きに重力ベクトルを与えた結果である．

【実験 3】時刻同期の動作確認

時刻同期機構により，ボーンアニメーションの再生が同期できているか確認する．実験には 2 台のクライアントを使用する．実験結果を図 9に示す．結果より正しくアニメーションの同期ができていることを確認した．

```
/* パーティクル発生 */
public void emit (){
 this.root.fire.particleNum = 1500;
 this.root.smoke.particleNum = 1500;
}
/* 色変更 */
public void changeColor(){
 Color blue = new Color( 0, 0, 255);
 this.root.fire.color = blue;
 this.root.smoke.color = blue;
}
/* 寿命変更 */
public void changeTime(){
 this.root.fire.endTime = 2000;
 this.root.smoke.endTime = 2000;
}
/* 重力変更 */
public void changeGravity(){
 Vector3 gravity = new Vector3(1.0, 1.0, 0.0);
 this.root.fire.gravity = gravity;
 this.root.smoke.gravity = gravity;
}
```

図 7 実験 2 で用いたスクリプトの一部



(a) (b)

(c) (d)

図 8 実験 2 の結果

図 9 実験 3 の結果

6. むすび

本稿では，我々が検討してきたモバイル MR システムのためのフレームワークにおけるアニメーションの表現力向上を目的とし，ボーンアニメーション及びパーティクル表現を用いたアニメーション制御機構を設計・実装した結果について述べた．実験の結果からアニメーション制御機構が正しく動作し，フレームワークの表現力が向上したと言える．

今後の展望として，表現可能なパーティクルの種類の拡充や，より実世界に即した描画方法の検討が挙げられる．

参考文献

[1] 柴田:“応用 1:モバイル AR 位置情報に基づく AR システム”, 情報処理, Vol. 51, No. 4, pp. 385 - 391, 2010.

[2] 山下他:“モバイル MR システム構築のための機能分散型フレームワーク-システムアーキテクチャとコンテンツ制御-”, 第 14 回日本バーチャルリアリティ学会大会論文集, 3A2-3, 2009.

[3] 縄谷他:“モバイル MR システム構築のための機能分散型フレームワーク(2)-コンテンツ制御機構の拡張-”, 第 15 回日本バーチャルリアリティ学会大会論文集, pp. 374 - 377, 2010.

[4] 前田他:“モバイル MR システム構築のための機能分散型フレームワーク(5)-サウンド制御機構の設計と実装-”, 第 16 回日本バーチャルリアリティ学会大会論文集, 14D-4, 2011.