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(4) —アプリケーション開発のためのオーサリングツールの設計と実装—

A Distributed Framework for Mobile Mixed Reality System (4)
- Design and Implementation of the Authoring Tool for Developing Applications -

辻愛二，柴田史久，木村朝子，田村秀行
Aiji Tsuji, Fumihisa Shibata, Asako Kimura and Hideyuki Tamura

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

Abstract: This paper describes an authoring tool which supports the process of application development based on the mobile mixed reality (MR) framework we proposed. Using our proposed framework, application developers (authors) can focus upon the application logic which controls the motions of the contents in MR space since position detection and rendering functions are provided by the framework. However, when the authors develop an MR application using this framework, they have to create the contents by combining images and 3D models which are previously prepared and have to register them to the database. Then, they develop the application by describing the motions of the contents using the original script language we proposed. However, such complicated work would sometimes induce mistakes because it takes much effort. Therefore, we propose the authoring tool which supports the application development process, namely contents creation, database registration, and script writing. Our authoring tool enables the authors to build an MR application in shorter period than before.

Key Words: *Mixed Reality, Framework, Authoring Tool*

1. はじめに

近年，モバイル端末の急激な性能向上を背景に，モバイル複合現実感（Mixed Reality; MR）システムへの期待が高まりつつある．然るに，システムの普及を念頭に置くと，個々のアプリケーションを個別に一から開発することは無駄が多い．そこで，筆者らはアプリケーション開発を効率化するフレームワークの開発を進めてきた[1].

提案フレームワークでは，MR に固有の機能とアプリケーション作成に必要な機能を分離することでアプリケーション開発の効率化を図っている．すなわち，MR に固有の機能をシステムレイヤとして予めアプリケーション開発者に提供し，アプリケーション開発者（以降，オーサ）はアプリケーション独自の部分として，提示する 3D モデルなどのコンテンツの動きのみをプログラムする，というスタイルを採用している．コンテンツの動きは，独自に設計したスクリプト言語[2]を用いて記述すればよく，アプリケーションに依存しないフレームワークを実現している．しかしながら，アプリケーションを開発するには，スクリプト記述や，スクリプトとコンテンツを結びつける作業など煩雑なものがあり，アプリケーション開発の効率を下げる要因となっていた．

そこで本研究では，筆者らの提案するモバイル MR システム構築のためのフレームワークにおいて，アプリケーション開発を支援し，効率化するためのオーサリングツールの開発について報告する．

2. 機能分散型モバイルフレームワーク

2.1 概要

既開発のフレームワークは，下記の 3 要件を重視して設計している．

- アプリケーションに非依存
- モバイル機器の種類や性能の差異を吸収可能
- 複数端末が同一の MR 空間を共有可能

これらの要件を満たすため，サーバ・クライアント型のアーキテクチャでシステム全体を設計した．サーバはアプリケーション全体を管理し，個々のコンテンツの動きをスクリプトエンジン[2]によって制御する．一方，クライアントはサーバから送られたコンテンツの位置姿勢情報をもとに，MR 提示を行う．オーサは，個々のコンテンツに対応する 3D モデルなどを準備した上で，それを制御するロジックをスクリプトとして記述することで，アプリケーションを作り上げる．そのため，MR を実現する上で必要な位置合わせやレンダリングなどの機能が再利用可能であり，生産性が向上している．

2.2 アプリケーション開発

2.2.1 アプリケーションの構成

本フレームワークで開発する MR アプリケーションは，3D モデルやテキストなどのコンテンツに対して，その動作を制御するためのスクリプトを外部に記述することで制御を行っている．このスクリプトをスクリプトエンジンで解析・実行することで MR アプリケーションを実現して

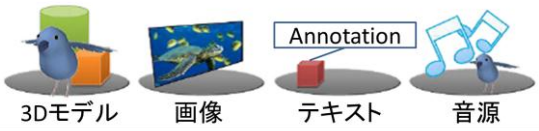

図 1　仮想オブジェクトの構成要素

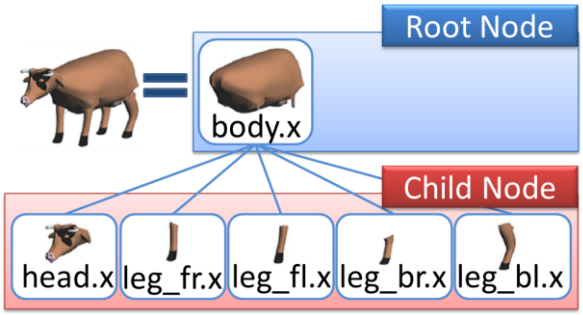

図 2　仮想オブジェクトの構成図

いる．以下にアプリケーション開発に必要な要素を述べる．

【コンテンツ】

アプリケーションで扱うコンテンツは MR 空間中に存在する仮想物体と実物体の 2 つに大別し，仮想オブジェクト，実オブジェクトと定義する．仮想オブジェクトは**図 1** に示すように，3D モデル・画像・テキスト情報・音源から構成される．3D モデルは X ファイル，メタセコイアファイルまたは，VRML により表現する．画像は，MR 空間中に配置する平面の仮想オブジェクトを指す．テキスト情報は，MR 空間中に提示する文字情報を指し，注釈情報として使用する．音源は，MR 空間内で一様に鳴る環境音，実オブジェクトを含むコンテンツから鳴る仮想音，そしてクライアントから発生し，インタラクションを行った時の効果音として扱うクライアント音を指す．仮想オブジェクトを構成するこれらの要素をコンポーネントと呼び，単体，または複数のコンポーネントをノードとした木構造によって 1 つの仮想オブジェクトを構成する．仮想オブジェクト同士の関係を記した木構造をシーングラフに定義する．仮想オブジェクトの具体例を**図 2** に示す．図 2 では，1 つの仮想オブジェクトが，1 つのルートノードに対して 5 つの子ノードを持つことで構成されている．

【スクリプト】

独自に設計したスクリプト言語は，記述が容易であり，かつ複雑なコンテンツの制御の表現が可能であることを目指して設計されている．そのため，個々のコンテンツの情報と動きを記述したプログラムを合わせて一つのオブジェクトとする，というオブジェクト指向の概念を導入している．具体的には，MR 提示する個々のコンテンツ毎に雛形となるクラスを定義し，そこにメソッドとしてコンテンツの動きを記述する．MR 空間を表現したクラスの中でこれらのインスタンスを生成することによって，コンテンツを配置する．このような枠組みにより，オーサは位置合わせやレンダリングなどの処理に煩わされることなく，コンテンツ制御のロジックのみに集中してプログラムが作

リスト 1　オブジェクトスクリプト

```
//Cow のメインクラスの継承
public class Cow extends Object{

    //フィールド                                          ← 初期化
    private double DEFULT_SCALE = 25
    int radius = 5;
    int pointx = 0;
    int pointy = 0;
    //コンストラクタ
    public Cow(Position3D pos,Orientation ori){
        this.pos = pos;
        this.ori = ori;
        this.scale = scale;
        this.presen = true;
        this.state = 0;
        //コンテンツの状態を初期化
        this.scale.x = DEFULT_SCALE;
        this.scale.y = DEFULT_SCALE;
        this.scale.z = DEFULT_SCALE;
        //コンポーネントの位置を初期化
        /*  省略  */
    }

    //メソッド追加部                                      ← コンテンツの動作
    void action(){
        CircleXY();
    }
    //XY 平面の円運動
    Void CircleXY( ){
        rad = rad + speed / 180.0 * Math.PI  * System.diffTime / 1000;
        //姿勢と位置を更新
        this.ori.z = 1.0 * (-90.0 + rad * 180.0 / Math.PI);
        this.pos.x = pointx + radius * Math.cos( rad );
        this.pos.y = pointy + radius * Math.sin( rad );
    }
}
```

リスト 2　ワールドスクリプト

```
//CattleMutilation メインクラスの継承
public class CattleMutilation extends MRWorld{

    //フィールド                                          ← 初期化
    //世界座標のオフセット
    private double OFFSET_X = 10300;
    private double OFFSET_Y = 10300;
    private double OFFSET_Z = 0;
    //ウシの出現上限数 5
    //コンストラクタ
    private int COW_NUM = 5;
    public CattleMutilation(){
        // Cow コンテンツを MRWorld に配置
        for(int i = 0; i < COW_NUM; i++){
            Position3D cowPos , cowOri;
            //初期位置の指定
            cowPos.x = OFFSET_X  + 100 * i;
            cowPos.y = OFFSET_Y  + 100 * I;
            cowPos.z = OFFSET_Z;
            //初期姿勢の指定
            cowOri.x = 0.0;
            cowOri.y = 0.0;
            cowOri.z = 0.0;
            //オブジェクトリスト追加
            _objectList.add(new Cow(cowPos , cowOri));
        }
    }

    //メインループ
    public void mainloop(){

        //インタラクションに対する制御                      ← クライアントからの入力の処理
        for (int i = 0; i < _clientList.size(); i++){
            Client client = _clientList.get(i);
            //Enter キーが押されたら，デバイスの位置に Cow を生成
            if( client.getInteraction() == System.INPUT.KEY_RETURN ){
                _objestList.add(new Cow(client.getDeviceA().getPosition()));
            }
        }

        //各コンテンツの動きを呼び出し                      ← コンテンツの動作の呼び出し
        for (int i = 0; i < _objectList.size(); i++){
            Object obj = _objectList.get(i);
            obj.action();
        }
    }
}
```

成可能となる．

ここで，コンテンツ毎の雛形となるクラスのスクリプトをオブジェクトスクリプト，MR 空間を表現しているクラスのスクリプトをワールドスクリプトと定義する．オブジェクトスクリプトは，変数を定義するフィールド，コンテンツの初期化を行うコンストラクタ，メソッド定義部からなる（**リスト 1**）．メソッド定義部には，MR 提示する個々のコンテンツ毎に雛形となるクラスを定義し，そこにメソッドとしてコンテンツの動作を記述する．ワールドスクリプトは使用する変数を定義するフィールド，コンストラクタ，メイン部からなる（**リスト 2**）．コンストラクタには，コンテンツの初期状態，インスタンス生成を記述する．メイン部には，インスタンス生成により MR 空間に配置され

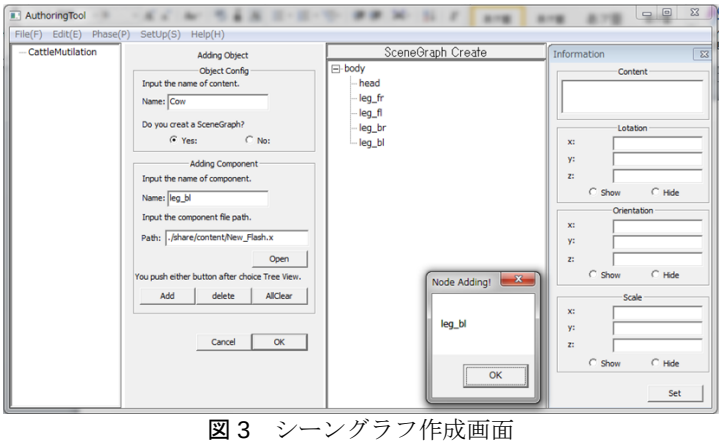

図 3　シーングラフ作成画面

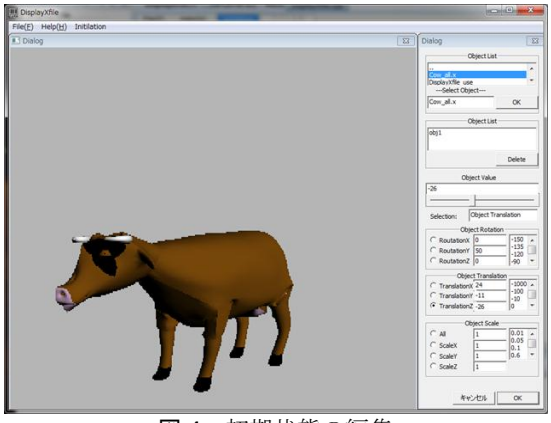

図 4　初期状態の編集

たコンテンツに対しての動作，またコンテンツ間の関係性を利用した制御や，クライアントからのインタラクションを契機とした制御などを記述する．

2.2.2 アプリケーション開発の手順と問題点

MR アプリケーションは下記の手順で開発する．

(1) コンポーネント作成
(2) コンテンツのシーングラフ作成
(3) オブジェクトスクリプト，ワールドスクリプト作成
(4) アプリケーションで扱う情報のデータベース登録
(5) オブジェクトスクリプト，ワールドスクリプト記述
(6) 開発したアプリケーションの動作確認

上記の手順でアプリケーションを開発する際に，これまでは下記の点が煩雑となり，ミスや作業遅延の原因となっていた．

- シーングラフの記述

シーングラフは XML で記述されており，オーサは XML を直接編集する．記述に誤りが存在した場合，提示するコンテンツが正しく提示されない．

- コンテンツのデータベース登録

作成したコンテンツはデータベースに登録する．このとき，オーサが直接 SQL を記述するため，登録するコンテンツが増加するに従って記述量が増加する．

- スクリプトの記述

各スクリプトにはフィールド，コンストラクタ，コンテンツの動作を記述するが，スクリプトを新たに作成するごとに共通の記述やメソッドを一から記述しなければならない．また，各コンテンツの配置の確認はアプリケーションが実行されるまで不可能である．

- 複数端末を使用したアプリケーションの確認

複数端末でアプリケーションが正しく実行されているかを確認するためには，複数の端末を用意する必要がある．しかしながら，用意できる端末の数には限りがあるため，多数の端末を利用したアプリケーションの確認が困難となる．

これらの処理をオーサが行うことは煩雑であり，アプリケーション開発の効率を下げる要因となっていた．そこで，オーサリングツールを開発し，上記の点を支援することでアプリケーション開発の効率化を図る．

3. オーサリングツール

3.1 概要

前章で述べた問題点を解決するため，オーサリングツールを設計・開発する．本稿で述べるオーサリングツールとは，フレームワーク上でのアプリケーション開発における，煩雑な処理を自動化するための支援ツールとして定義する．本章では，提案ツールの設計方針，オーサリングの流れについて述べる．

3.2 設計方針

前章で述べたアプリケーション開発の手順に従って，オーサリングツールを設計する．アプリケーションは，(1) コンテンツの作成，(2) スクリプトの記述，の 2 つを繰り返すことで行う．そこで，(1)の処理を準備フェーズ，(2)の処理を開発フェーズと定義する．準備フェーズで作成したコンテンツに対して，開発フェーズで各スクリプトの記述を支援する設計を取ることで，実際のアプリケーション開発の手順に即した支援が可能となる．準備フェーズでは下記の機能によってコンテンツの作成を支援する．

- コンテンツ生成
- データベース登録

開発フェーズでは，下記の機能によってスクリプトの記述を支援する．

- 初期状態の編集
- メソッドのテンプレート
- プレビュー

また，本ツールを用いて開発したアプリケーションは後述するプロジェクトという単位で管理する．

3.3 オーサリングの流れ

3.3.1 準備フェーズ

(1) コンテンツ作成

本機能では，コンテンツとコンポーネントの名前の設定及び，使用する画像や 3D モデルファイルの登録を行う．操作画面を**図 3** に示す．情報の登録には，図 3 左部のパネルで入力する．ここではコンテンツ情報として，コンテンツ名，コンポーネント名，使用するファイルの 3 つを登録する．登録された情報は，図 3 中央に示すような，コンポーネントをノードとした木構造で表現される．オーサは各ノードに対してコンポーネントの追加・編集を行う．また，

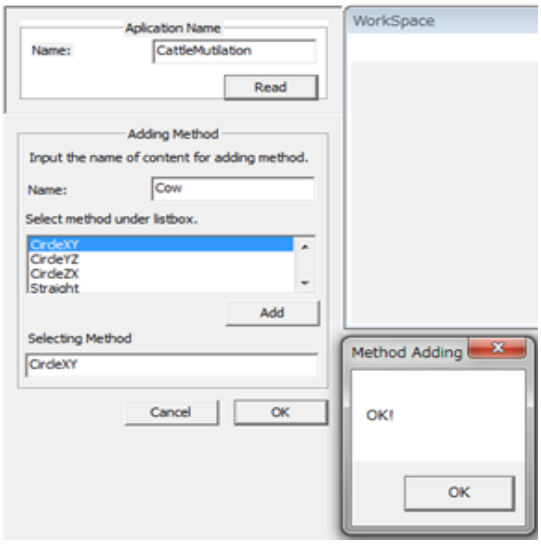

図 5　メソッド追加画面

コンテンツの作成が完了したとき，スクリプトの雛形が同時に生成される．

(2)　データベース登録

コンテンツは，ツール上で入力されたコンテンツ情報に基づいて，データベースに自動的に登録される．予め入力した情報を登録することで，オーサが直接データベースに登録する際に発生する記述の誤りを無くすことができる．

3.3.2 開発フェーズ

(1)　初期状態の編集

コンテンツの初期状態を VR 画面上で編集することで，アプリケーションを実行する前の段階での位置関係を考慮したコンテンツの配置を実現する．コンテンツの配置には，各コンポーネントの位置・姿勢・スケールを編集することで行う．この機能によって，アプリケーションを実行せずにツール上で予めコンテンツの配置を確認しながら調整することが可能となる（**図 4**）．調整には，図 4 右部の操作パネルを使用する．選択したコンテンツに対して，位置・姿勢・スケールを変更することで調整する．

また，MR 画面上での初期状態の配置を行う．MR 画面上でのコンテンツの配置は，実空間での利用を考慮し，実オブジェクトと仮想オブジェクトとの位置関係を確認しながら調整する．VR 画面上で配置した結果にあわせて，MR 画面上で調整することで，より詳細な配置が可能となる．

編集した内容はワールドスクリプト中のコンストラクタに反映される．

(2)　メソッドのテンプレート

使用する頻度の高いメソッドを予め提供することで，開発の効率化を図る．利用画面を**図 5** に示す．ツール上では，メソッドを追加したいコンテンツを選択する．そして，追加するメソッドを選択する．追加されたメソッドは，オブジェクトスクリプトのメソッド定義部に記述される．提供するメソッドの例を**表 1** に示す．

また，ツール開発者が予め提供するメソッドだけでなく，オーサが記述したメソッドを自由にテンプレートとして登録できる設計とする．このような設計を取ることで，オーサは柔軟にアプリケーションを開発することが可能となる．

表 1　メソッドテンプレートの例

メソッド名	処理内容	入力情報
CircleXY	XY 平面に対する円運動	中心座標，半径
CircleYZ	YZ 平面に対する円運動	中心座標，半径
CircleZX	XY 平面に対する円運動	中心座標，半径
SineCurveXY	XY 平面に対するサインカーブ	振幅，周期
SineCurveYX	YZ 平面に対するサインカーブ	振幅，周期
SineCurveZX	ZX 平面に対するサインカーブ	振幅，周期
Straight	始点と終点に対する直線運動	始点，終点

(3)　プレビュー

プレビューでは，開発したアプリケーションの確認を行う．支援する機能として，仮想クライアント機能とコンテンツの情報提示機能を開発した．

仮想クライアントは，実際に端末を用意する代わりに，仮想的なクライアントを任意の位置に配置することで，複数端末を利用した場合のアプリケーションの動作を確認する機能である．本機能によって，端末を用意すること無く，複数視点でのアプリケーションの確認，インタラクションが実現される．

コンテンツの情報提示機能では，コンテンツの位置・姿勢・スケールを表示する．アプリケーション実行時のコンテンツの情報を提示することで，コンテンツが意図した動作をしているか確認することができる．

(4)　プロジェクト管理

ツール上で作成したアプリケーションは，プロジェクトという単位で管理する．アプリケーション 1 つに 1 つのプロジェクトを持つ．プロジェクトはアプリケーションに必要なコンポーネント，スクリプト，シーングラフ及び，ツールで設定した情報が格納される．このようにすることで，複数のアプリケーションを開発した場合にも管理が容易となる．

4.　まとめ

本稿では，モバイル MR システム構築のためのフレームワークにおける，オーサリングツールを開発について報告した．本ツールは，アプリケーション開発において煩雑な作業となるシーングラフの記述，コンテンツのデータベース登録，スクリプトの記述，および複数端末を使用したアプリケーションの確認作業を支援する．オーサは，本ツールを使用することで，従来に比べて容易にアプリケーションを開発することが出来る．

今後は，ユーザビリティの向上を目指すとともに，フレームワークの改良にあわせてツールの機能拡張を行う．

参考文献

[1]山下智紀 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機構-"，第 14 回日本バーチャルリアリティ学会大会論文集，3A2-3，(2009.9)

[2]縄谷侑司 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-"，本大会，2010