Vol.2012-EC-23 No.20
2012/3/27

MikuMikuDance のコンテンツを Unity で動作させる方式の提案

Eiichi Takebuchi[†1]　Akira Hattori[†1]　Haruo Hayami[†1]

本研究では MikuMikuDance で利用されている PMD 形式及び VMD 形式を Unity 上にて動作させることに成功した。これにより、Web 上で公開されている多数のフリーで公開されている 3DCG モデルデータやアニメーションデータをゲーム開発に活かすことが可能となる。本論文ではその動作方式を試作システムにて実装を行った際の手法を述べる。

The method of emuration for MikuMikuDance's contents on Unity

We describe an emulation method to use MikuMikuDance's contents as the PMD and VMD file formats on Unity. By using our system includiong this method, users would have an game development that utilizes more free 3DCG models and animations contents on the Web. We explain how to enable the method.

1. はじめに

Unity**??**とは 3DCG ゲームを開発するためのゲームエンジンである。ゲームワールドを構築するのに必要なデータを視覚的に配置でき、コンパイルなしでもその場でゲームワールドの様子を確認できることが特徴である。プログラムの多くはスクリプトで作成することが可能であり、物理エンジンやライティングなども自動で処理されるため、プログラミング初心者などでも高度な視覚表現のあるゲームを開発することが可能となっている。

MikuMikuDance(以下 MMD)**??**とは 3DCG アニメーションソフトウェアの一つであり、フリーウェアとして公開されている。アニメーションを行うために必要な最小限のインターフェースが特徴であり、3DCG の初心者でも比較的簡単に 3DCG アニメーションを作成することができる。また、MMD のコミュニティも巨大である**??**。多くの 3DCG モデル製作者が 3DCG モデルデータを公開している。この 3DCG モデルデータの多くはライセンスフリーであり、コンテンツを作る際に制限を設けていないことが多い。このため、3DCG モデリングスキルのない初心者でも高い品質の 3DCG モデルデータを利用してコンテンツを作成することが可能であり、現在もニコニコ動画などを中心として動画制作などのコンテンツ制作の場が培われている。

本研究ではこの 2 点に着目し、Unity 上で MMD で公開されているコンテンツを利用することが可能であれば、3DCG モデリングに必要なスキルがなくとも高い品質のゲームが開発可能となるのではないかと考えた。特にプログラミングスキルと 3DCG モデリングやアニメーションなどのデザインスキルは平行して勉強するのには長い年月が必要となる。デザインスキルを既に公開されているコンテンツで補うことによって、趣味としてのゲーム開発を促進したい目的で本研究を行った。

本研究で提案する手法は、MMD における 3DCG モデルデータを表現する PMD 形式とアニメーションデータを表現する VMD 形式を Unity で読み込み、Unity 内における各種データ構造を構築し、動作させることによって実現する。

2. 研究背景

個人によるゲーム開発は非常に多くのスキルが必要となる。プログラミングスキルはもちろんであるが、ゲームワールドを構築するための 3DCG モデルを作成するスキルも必要である。本来ならばプログラマはプログラムを書くべきであって、3DCG モデルを作成するためのスキルやテクスチャを描くためのデッサンなど、デザインスキルを磨くことはあまり好ましいとは言えない。必ずしもプログラマにとってデザインが楽しいと感じるとは限らないからである。

また、プログラマから見ても高度な数学や物理、シェーダなどの要素技術が楽しいと感じるとは限らず、ゲーム開発からそれらをすぐに思い浮かべる人は非常に少ない。

そこでプログラマがゲームデザインに注力するための良い方法としてゲームエンジンを利用することが挙げられる。ゲームエンジンであればゲーム開発に必要な高度な要素技術が

†1 神奈川工科大学
Kanagawa Institute of Technology

© 2012 Information Processing Society of Japan

既に実装・自動化され、スクリプトなどにより簡易的に操作が可能なのである。

一方でデザインスキルを解決する方法としては、フリーで公開されている素材を利用することである。個人におけるゲーム開発においてフリー素材を利用することは珍しくない。RPG ツクールであればマップチップの画像ファイルが多く公開され、ライセンスも緩いためゲーム開発に利用することが可能である。

しかし、3DCG モデルデータの場合、フリーで公開されているデータは少ない。Google 3D Gallery では多くの 3DCG モデルデータが公開されているが、Google Sketchup でしか開くことができないため、ライセンスが自由であるのにも関わらずゲームへの転用は難しい。特にゲーム開発において必要なのはゲームにおける主人公を表すための人型などの 3DCG モデルデータである。

本研究においてまず初めに着眼したのは MMD である。MMD のコミュニティにおいて数多くの 3DCG モデルデータが公開されている。VOCALOID をはじめ、東方 Project などの 3DCG モデルデータを中心にニコニコ動画においてその作品が多く投稿されている。MMD 利用者は公開されている 3DCG モデルデータを利用し、動画の制作を行い投稿している。その際、MMD 利用者に必要なのはアニメーションスキルであり、3DCG モデリングスキルではない。

そこで MMD におけるコミュニティ性に着目した。MMD で公開されている 3DCG モデルデータをゲーム開発において利用することが可能であれば、プログラマにデザインスキルが問われることなく、高い品質で視覚表現を行うことが可能となるのである。本研究が実現すればプログラマがデザインを行うような無理が起こらなくなり、自分の好きなキャラクタなどを利用してゲーム開発を行うことができるようになるのである。

本研究ではゲーム開発を行うプラットフォームとしてゲームエンジンの一つである Unity を利用した。他のゲームエンジンと比べインターフェースが洗練されて扱い易いからである。本研究における試作システムは Unity のプラグインとして実装を行った。

3. 利 用 技 術

本節では試作システムの実装において利用した技術及びソフトウェアについて説明する。

3.1 Unity

Unity とは、Unity Technologies 社が開発しているゲームエンジンの一つである。試作システムは UnityEngine を利用してプラグインとして実装を行った。

Unity の特徴はゲームワールドの構築やゲームデザインを GUI 上で簡単に行える点である。インターフェースも既存の 3DCG ソフトウェアと似ているため、デザイナーでも比較的容易に開発に参加することが可能となっている。また、物理エンジンやライティングなども自動的に行われるため、プログラマがこれらをゼロから開発する必要がない。

GUI で行えないゲームデザインの開発では主に Javascript や C#などのようなスクリプト言語を用いて開発を行う。従来ではメモリ管理やラップされたオブジェクト指向の概念、ハードウェアへの理解など、ゲーム開発を行うのに必要な知識は膨大であったが、ゲームエンジンによって大部分がラップされているため高度なプログラミングスキルがなくともゲーム開発が可能である。

3.2 MikuMikuDance

MikuMikuDance（MMD）とは、表示されている 3DCG モデルデータにアニメーションを行うためのソフトウェアである。樋口優氏が開発を行い、フリーで公開されている。ソフトウェアの導入時からいくつかの 3DCG モデルデータが用意され、初心者でもすぐにアニメーションを体験できることが大きな特徴である。

他のソフトウェアとの大きな違いは、キャラクターをすぐにアニメーションさせることができる点である。通常の 3DCG ソフトウェアでは 3DCG モデルデータからセットアップまで全て個人もしくはチームで行わなければならないが、MMD では既にいくつかの 3DCG モデルデータが用意されているだけでなく、多くの 3DCG モデルデータがフリーとして公開されているため、3DCG を動かしたことのない初心者でも 3DCG モデリングやテクスチャリングなどのスキルを必要とせず、読み込ませるだけですぐに高品質で好みのキャラクターを動かすことができるのである。

ソフトウェアの外観を図 1 に示す。

4. 試作システムの実装

本研究における試作システムとして、PMD 形式の変換を行う PMD Loader と VMD 形式の変換を行う VMD Loader を実装した。PMD Loader では PMD 形式のファイルを読み込み、Unity 上で 3DCG モデルデータの構成に必要なデータの変換を行う。VMD Loader では VMD 形式のファイルを読み込み、PMD Loader で変換された 3DCG モデルデータのアニメーションに必要なデータの変換を行う。

試作システムの構成を図 2 に示す。

インターフェース部によってシステムの実行が知らされると処理実行部にその通知が送られる。その通知を基に読み込み部がまず呼び出される。読み込み部では PMD/VMD ファイ

© 2012 Information Processing Society of Japan

Vol.2012-EC-23 No.20
2012/3/27

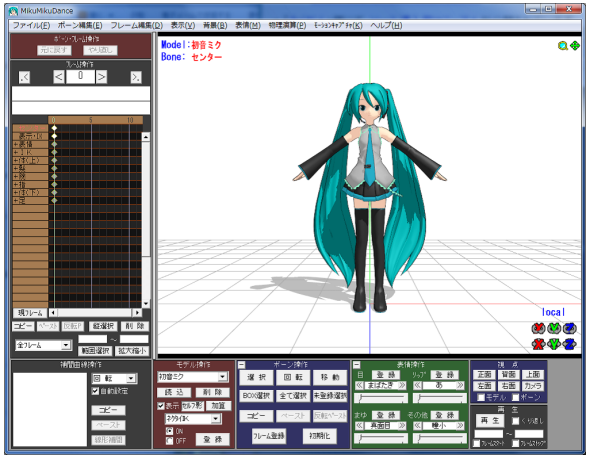

図 1 MMD の外観

ルの読み込みが行われ、それぞれ中間フォーマットへ書き込みが行われる。中間フォーマットは C#におけるデータ構造であり、class 型として宣言されている。中間フォーマットの作成が完了すると変換部が呼び出される。変換部では中間フォーマットを基に Unity における各種データ構造へ書き込みが行われる。Unity における適切なデータ構造の構築が終わるとそのデータ構造を Project へ登録を行う。Project への登録が行われると Unity のゲームワールドで変換したデータを利用することができるようになる。

4.1 インターフェース部の実装

本項では実装を行った試作システムのインターフェース及び操作方法について述べる。

PMD Loader のインターフェースは図 3 の通りである。PMD ファイルを PMD File にドラッグ＆ドロップし、Convert ボタンの押すことで変換が開始される。変換直後はエディタ上で図 4 のように表示される。

変換後には PMD ファイルと同じディレクトリに以下のファイル及びディレクトリが追加される。

- Material ディレクトリ
- Physics ディレクトリ
- Asset ファイル
- Prefab ファイル

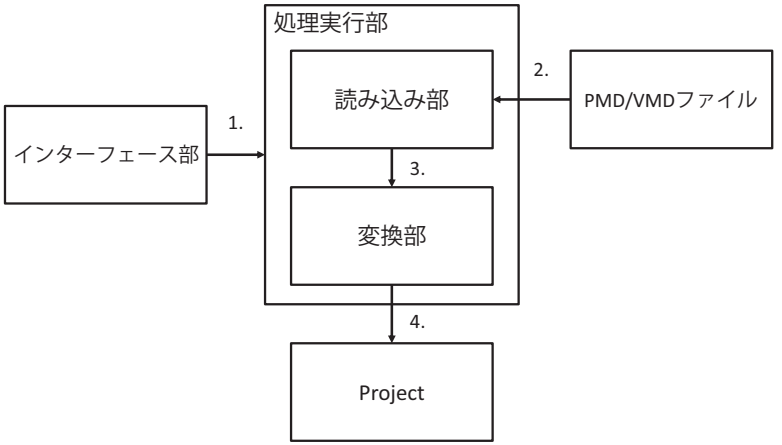

図 2 試作システムの構成

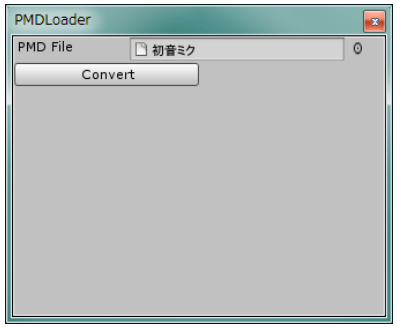

図 3 PMD Loader のインターフェース

© 2012 Information Processing Society of Japan

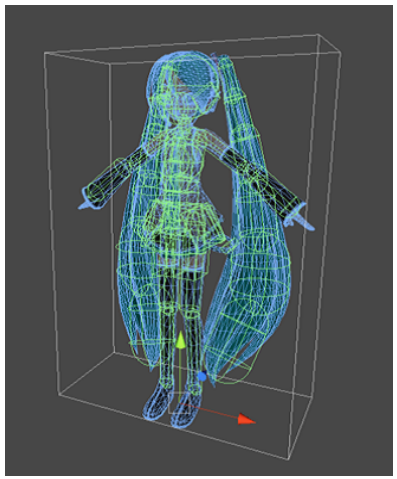

図 4 変換直後にエディタ上に表示されるデータ

Material ディレクトリには、PMD ファイルから抜き出した材質の情報を Unity における Material データに変換した Asset ファイルが追加される。材質にはその名前が割り当てられていないため、PMD ファイルの「ファイル名＋番号」として格納される。

Physics ディレクトリには、PMD ファイルから抜き出した剛体の情報を Unity における PhysicMaterial データに変換した Asset ファイルが追加される。ファイル名の命名法については Material ディレクトリと同様である。

PMD ファイルと同ディレクトリに保存される Asset ファイルには 3DCG モデルデータを表すためのデータが保存される。このファイル単体では Unity 上で扱うことはできない。

Prefab ファイルは 3DCG モデルデータである Asset ファイルと Material ディレクトリ内にある Asset ファイル、Physics ディレクトリ内にある Asset ファイルの参照を表したものである。3DCG モデルデータを表す Asset ファイル単体ではゲームワールドに存在させることができなかったが、Prefab 化した上で各種必要なデータ構造を割り当てることによってゲームワールドで扱うことができるようになる。PMD Loader で変換を行った Prefab ファイルは既に各種必要なデータ構造の割り当ては完了しているため、すぐにゲームワールド内にて利用することができる。

VMD Loader のインターフェースは図 5 の通りである。PMD Loader で生成された Prefab ファイルを PMD Prefab に、変換を行いたい VMD ファイルを VMD File にドラッグ＆ドロップする。

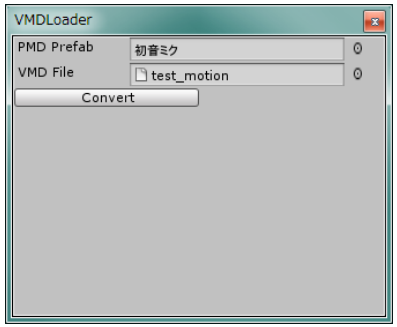

図 5 VMD Loader のインターフェース

変換後に PMD Prefab にセットした Prefab ファイルに VMD ファイルと同名の Animation Clip が追加される。Animation Clip が追加されたことの確認は Project 内で確認することができる。

4.2 読み込み部の実装

読み込み部では PMD 及び VMD ファイルの読み込みを行い、中間フォーマットへの書き込みも行う。直接読み込み変換することも可能であるが、試作システムではオープンソースで公開を前提として実装を行ったため、汎用性を考慮して中間フォーマットを作成した。

中間フォーマットのクラス名及びその概要を PMD 形式??は表 1、VMD 形式??は表 2 に示す。

4.3 変換部の実装

変換部では読み込み部で生成した中間フォーマットから Unity が持つ各種データ形式への変換を行う。変換を行い格納する先のデータ構造については表 3 に示す。

PMD 及び VMD ファイルと Unity のデータ構造に関する関連は図 6 及び図 7 に示す。

5. 評 価 実 験

試作システムは実装後にオープンソースで公開を行った。評価実験協力者はダウンロードして利用している方から募り、Web でのアンケート方式にて回答を頂いた。詳細な条件は

© 2012 Information Processing Society of Japan

Vol.2012-EC-23 No.20
2012/3/27

表 1 PMD 形式の中間フォーマットのクラス一覧

クラス名	概要
Header	PMD ファイルのヘッダー情報
VertexList	頂点の総数を記したクラス
Vertex	頂点の情報
FaceVertexList	ポリゴンを構成する頂点の情報
MaterialList	材質の総数を記したクラス
Material	材質の情報
BoneList	ボーンの総数を記したクラス
Bone	ボーンの情報
IKList	IK の総数を記したクラス
IK	IK の情報
SkinList	表情の総数を記したクラス
SkinData	表情を構成する頂点の情報
SkinVertexData	対象頂点がモーフした後の頂点座標を記したクラス
SkinNameList	表情枠に表示する表情の名前を記したクラス
BoneNameList	ボーン枠の名前を記したクラス
BoneDisplayList	ボーン枠に表示するボーンの総数を記したクラス
BoneDisplay	ボーン枠に表示するボーンの名前を記したクラス
EnglishHeader	英語用のヘッダー情報
EnglishBoneNameList	英語用のボーン枠の名前を記したクラス
EnglishSkinNameList	英語用の表情枠に表示する表情の名前を記したクラス
EnglishBoneDisplayList	英語用のボーン枠に表示するボーンの名前を記したクラス
ToonTextureList	トゥーンテクスチャのパスを記したクラス
RigidBodyList	剛体の総数を記したクラス
RigidBody	剛体の情報
JointList	Joint の総数を記したクラス
Joint	Joint の情報

表 2 VMD 形式の中間フォーマットのクラス一覧

クラス名	概要
Header	ヘッダー情報
MotionList	モーションキーフレームの総数を記したクラス
Motion	あるキーフレームにおけるボーンの情報
SkinList	表情キーフレームのの総数を記したクラス
SkinData	あるキーフレームにおける表情の情報
CameraList	あるキーフレームにおけるカメラの情報
CameraData	カメラキーフレームにおけるカメラの情報
LightList	あるキーフレームにおけるライトの情報
LightData	ライトキーフレームにおけるライトの情報
SelfShadowList	あるキーフレームにおけるセルフシャドウの情報
SelfShadowData	セルフシャドウキーフレームにおけるセルフシャドウの情報

表 3 Unity におけるデータ構造を表すクラス一覧

クラス名	概要
Mesh	ポリゴンを表すクラス
Material	材質を表すクラス
Texture	テクスチャを表すクラス
SkinnedMeshRenderer	アニメーションを行うためのレンダラー
GameObject	ゲームワールド上のオブジェクトを表すクラス
Transform	ゲームワールド上での座標や姿勢、親子関係を表すクラス
Collider	当たり判定を行うためのクラス
Rigidbody	剛体による物理演算を行うためのクラス
PhysicsMaterial	剛体の摩擦力などを表したクラス
Joint	ボーンとボーンを拘束するためのクラス
Animation	アニメーションの動作をまとめるクラス
AnimationClip	キーフレームをまとめた単一のアニメーションを表したクラス
Keyframe	キーフレームごとの情報を表したクラス
AnimationCurve	キーフレームにおける補間曲線を表したクラス

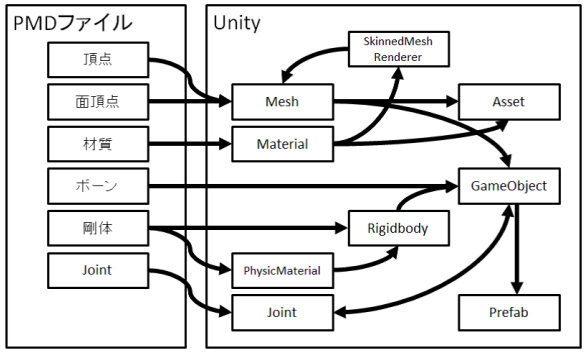

図 6 PMD ファイルと Unity のデータ構造の関連

© 2012 Information Processing Society of Japan

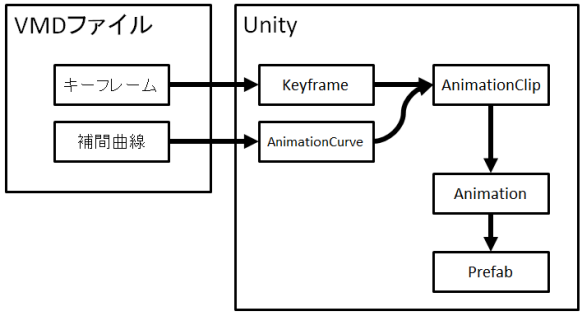

図 7 VMD ファイルと Unity のデータ構造の関連

以下の通りである。

- Web でのアンケート方式
- ダウンロード数は 1063 件（2012 年 2 月 27 日現在）
- 評価実験協力者はその中から 12 名
- 評価実験項目は 5 段階評価で 3 種類、自由記述回答は 2 種類

評価実験項目は以下の通りである。

(1) 試作システムは使いやすかったか
(2) 試作システムを利用して以前と比べて開発しやすくなったか
(3) 試作システムの全体的な評価
(4) 3 を選んだ理由
(5) 自由記述

評価実験の結果は表 4 の通りである。

表 4 評価実験の結果

評価実験項目	1 ⟷ 5					評価の割合 (%)
試作システムは使いやすかったか	0	1	5	4	2	67
試作システムを利用して以前と比べて開発しやすくなったか	0	1	4	4	3	59
試作システムの全体的な評価	0	0	0	5	7	100

評価の割合では 1 から 5 までの評価のうち 5 と 4 の評価について集計を行った。

評価実験の結果、全体的な評価が非常に高い結果となった。しかし、評価実験項目 1 及び 2 の 2 点に関しては約半数の評価実験協力者が評価を行っていない。

また、評価実験項目 4 及び 5 に関しては、研究に対して高い評価を行っている記述が多く見られた。評価実験項目 4 では 7 名の評価実験協力者が肯定的な意見を回答し、評価実験項目 5 では 3 名の評価実験協力者が肯定的な意見を回答していた。ただし、それぞれ研究の将来に対する考察や意見などが多く見られた。

全体的な評価に対して実際の試作システムに対する評価が低い。肯定的な意見も試作システムではなく研究の将来を評価する回答が多かったため、試作システムは将来性があるものの現段階ではまだ有用性がないと考えられる。ただし、研究への取り組みに対しては評価を頂いているため、研究を継続し、発展させることが可能であれば有用性を見いだせられると考えられる。

6. おわりに

評価実験から本研究の有用性を確認することができた。

現在、試作システムはオープンソースで公開している。今後は開発の継続だけでなく、ドキュメントの整備を行うことによって利用者への導線を敷きたいと考えている。

参考文献

1) 樋口優：Vocaloid Promotion Video Project（オンライン），
入手先⟨http://www.geocities.jp/higuchuu4/⟩（参照 2012-02-28）.
2) Unity Technologies : 3D Game Engine (online), Unity Technologies,
available from ⟨http://unity3d.com/⟩ (accessed 2012-02-28).
3) VPVP wiki：モデルデータ, VPVP wiki（オンライン），
入手先⟨http://www6.atwiki.jp/vpvpwiki/⟩（参照 2012-02-28）.
4) t_tetosuki：「MMD のモーションデータデータ (PMD) 形式　めも」通りすがりの記憶（オンライン），入手先
⟨http://blog.goo.ne.jp/torisu_tetosuki/e/209ad341d3ece2b1b4df24abf619d6e4⟩
（参照 2012-02-28）
5) t_tetosuki：「MMD のモデルデータ (PMD) 形式　メモ　(まとめ)」通りすがりの記憶（オンライン），入手先
⟨http://blog.goo.ne.jp/torisu_tetosuki/e/bc9f1c4d597341b394bd02b64597499d⟩
（参照 2012-02-28）
6) Unity Technologies : Unity Script Reference (online), Unity Technologies, available from ⟨http://unity3d.com/support/documentation/ScriptReference/index.html⟩
(accessed 2012-02-28).

© 2012 Information Processing Society of Japan