

平成 28 年 1 月 22 日

慶應義塾大学 理工学研究科

開放環境科学専攻 情報工学専修 御中

## 修士論文発表における質問に対する回答文

著者学籍番号：81621728

題目：

音源に同期する運指および表情に注目した吹奏アニメーションの自動生成

拝啓

時下ますますご清祥のこととお慶び申し上げます。

この度は、私の修士論文発表において貴重なコメントを賜りまして誠にありがとうございました。頂戴したコメントに従い、以下のような方針に基づいて改良を加えましたので、ご確認を何卒よろしくお願い申し上げます。

敬具

堀井 絵里

質問 1 不自然な例として示した手描きのアニメーションと、自動生成している 3DCG のアニメーションは本質的に違うのではないか。(斎藤英雄先生)

仰る通り、本質は違います。しかし、実際に 3DCG でそのようなシーンを再現する場合も、音と動きを 1 つずつ合わせるには時間と労力がかかるということは、アニメーション制作に詳しい者から聞いております。

実際に、本手法によるアニメーション制作の時間および労力の軽減について、アニメーション制作に詳しい者へアンケートをとったところ、実際に軽減されると予想した者がほとんどでした。そのため、本手法により 3DCG のアニメーション制作の時間および労力の軽減が可能であるといえます。こちらのアンケート結果は、4.2.4 項に示しております。

質問 2 音と指のモーションはどのようにマッピングしているのか。(斎藤英雄先生)

トランペットに関しては、それぞれの指の状態が、ピストンを押すか押さないかの 2 つとなります。一方トロンボーンに関しては、スライド管を止める場所が 7箇所存在します。いずれの楽器も、音と指や腕の状態を 1 対 1 で対応させることができるので、音が変わるとその音が鳴るような姿勢に遷移し、それをその長さだけ保つ、といった実装になっております。

上記の内容は、3.8.1 項にて詳説しております。

質問 3 金管楽器ごとにマップ表を作成することにより、どの楽器でも自動生成が可能であるという点が新規性なのか。(斎藤英雄先生)

はい、仰る通りです。マップ表とモデルさえあれば、どの金管楽器でも容易に吹奏アニメーションの自動生成が可能となっております。

こちらについては、2.5 節に記載いたしました。

質問 4 弦楽器に拡張する時に解決が必要となる課題は何か。(杉浦先生)

弦楽器は金管楽器とは異なり表情の変化は小さいですが、音と弦の押さえ方が 1 対 1 ではないため、どの指使いが適切なのかを、音を鳴らす度に考慮するような処理を加える必要があります。

弦楽器の場合は 2.1 節で紹介している先行研究が存在するため、そちらも参考に実装を行うことにより、対応が可能であると考えられます。

#### 質問 5 高音を吹き続けると辛くなる様子を、呼吸のパラメタを用いて表現することについて。(杉浦先生)

頂きましたアドバイスの通り、一定時間ブレスをしていない、あるいは高音域が続くなど、辛くなるような条件を満たした際に辛そうな表情をする、といったことは実現が可能です。こちらに関しては、モーションに関するメタデータを用意し、そちらに表情やモーションの指定を記載することを考えております。自動的に算出できる場合は算出し、算出ができない場合はそちらのデータに記載することにより、表現が可能であると考えられます。

発表の場で申した通り、表情は口元のみ制御していますが、そのためには、メッシュを制御する必要があります。Unreal Engine でメッシュを制御する際、モーフターゲットという機能を用いますが、この機能はブレンドシェイプが適用されていないメッシュには使用できません。ここで、ブレンドシェイプとは、あらかじめ用意された複数の表情モデルをパラメタの調整により組み合わせることで、さまざまな表情を作成するアニメーション手法をさします。今回用いたユニティちゃんは、デフォルトで口元にブレンドシェイプが設定されているため、モーフターゲットによる口元の制御は可能です。しかし辛い表情などを再現するためには、デフォルトの設定ではブレンドシェイプが不足しているため、他のソフトウェアを介して設定する必要があると考えられます。つまり、Unreal Engine のみでは再現が不可能であるといえます。

口元の制御方法に関しては 3.8.2 項、口元以外を制御する際の制約については、今後の課題として 5.2.1 項にまとめました。

#### 質問 6 人による動きの違いを各演奏者に適用する方法について。(杉本先生)

元となるモーションは全て同じですが、そのモーションの大きさを 0 から 1 のパラメタでランダムに設定することにより、個人差を出す工夫をしています。杉浦先生からご指摘がありました、演奏者ごとに体型が異なる場合は、質問 5 で述べたメタデータにその旨を記載し、モーションを分ける必要があると考えられます。

前者の内容については 3.8.3 項に、後者の内容については今後の課題として 5.2.4 項に記載いたしました。

#### 質問 7 ユニティちゃんを Unreal Engine で使用することについて。(杉本先生)

著作権的には問題ございません。こちらについては『ユニティちゃんライセンス条項』の第 3 条にて判断できます。(URL: [http://unity-chan.com/contents/license\\_jp/](http://unity-chan.com/contents/license_jp/))

上記の内容は、3.6 節に明記いたしました。

以上

修士論文

2017年度

音源に同期する運指および表情に注目した 吹奏アニメーションの自動生成

堀井 紘里

(学籍番号 : 81621728)

指導教員 教授 藤代 一成

2018年3月

慶應義塾大学大学院理工学研究科  
開放環境科学専攻

## 論文要旨

音楽演奏を題材としたアニメーションは多く存在する。そこでは、実際に演奏する演奏者の身体の動きに近い演奏シーンが生成されている。しかし、楽器を演奏するキャラクタの運指に注目すると、音楽と指の動きが完全に同期されていないことがあり、強い違和感を感じる。また、完全に同期されているアニメーションを作成するには、多くの労力が必要となる。そこで本研究では、管楽器を演奏するキャラクタの吹奏アニメーションを、音楽に容易に同期させるため、音源から自動的に運指のアニメーションを生成することを目指す。より具体的には、ウインドシンセサイザを用いて作成したMIDI音源を解析することにより、運指が音源に同期した、自然な吹奏アニメーションを実現する。

近年、コンピュータの処理速度が向上したことで、高品質な画像の高速な描画が可能となった。この影響により、ゲームや映像をユーザが完成版に近い品質で編集できる技術が発達している。このようなCG技術の台頭やITの発達で、最新のコンピュータ上に仮想の社会や人間をつくる技術が研究者やCG関係者の注目を集めている。しかし人間のパーツの多くは目や肌、髪の毛のようにやわらかい、小さい、半透明といった特徴をもつパーツで構成され、コンピュータには簡単に描画できないものが多い。特に頭髪は、人間にとて重要なパーツでありながら、その複雑さが原因で高速な描画が難しい物体として知られている。また頭髪状物体が落とす影も同様に、一般的な描画手法を使うとちらつきやありえない濃度の影が生じるため、計算が難しい描画対象として知られている。

頭髪は半透明で、かつ細長い円筒の集合であり、このような幾何形状をもつ物体は頭髪の形状、位置や視点、照明に変化がある場合、影を描画する際にちらつきが生じる。このちらつきのうち、頭髪が半透明で遮蔽判定が難しいことによるちらつきは、影を描く場所に光が届くまでに通過する頭髪密度の積算値を使い影を描くことで解決できる。一方、髪が細すぎてコンピュータが髪を認識できないことによるちらつきは、投影後にちらつきが生じている部分と周囲の色を平均化して抑制してきた。しかしこの平均化処理は、結果の品質が投影後の単位面積あたりの画素数に依存してしまう。

そこで本論文では、頭髪の密度を積算する既存手法を拡張して、大きさが可変で不透明度分布をもつ正方形小板で頭髪の三次元形状を近似することにより、結果の品質が画素数に依存しないアンチエイリアシング手法を提案する。これに加えて既存研究では積極的に行われてこなかった、出力画像の定量的な品質評価をするために新たな評価方法を作成した。この評価方法を使用して既存手法と提案手法の出力画像の品質を比較した結果、提案手法は計算速度を大きく減少させることなく効果的にちらつきを抑制できることを確認した。

### キーワード

アニメーション、吹奏楽、金管楽器、自動生成。

## Thesis Abstract

### An Object-Space Anti-Aliasing Method for High-Quality Shadowing of Hair-Shaped Objects

Recent improvement in CPU and GPU performance made it possible to render high-quality computer graphics in real time. Accordingly, technologies to edit video games or movies with nearly-final quality previews have been developed. With support of such rapid growth of CG technology as well as other IT developments, it has begun attracting much attention from related researchers and practitioners to construct realistic and dynamic virtual societies and humans on a modern computer. However, making virtual humans is a very difficult task because they are made up of many soft, small or transparent parts such as skin, **hairs** or eyes. Especially, human hairs are well-known as intricate objects which require much executing time to render and cause **aliasing**, even though they are essential parts of humans. Shadows cast from hairs are also known as difficult objects to render because they generate aliasing and too far dense shadows with commonly-used shadowing methods.

Hairs are **semi-transparent** and can be presented as mass of thin cylinders, but this style of geometric modeling causes aliasing to rendered shadows when users update a shape or a position of hair, camera angle and/or the position of lights. As for a type of aliasing caused by difficulty to detect occlusion due to the transparency of hairs, even previous shadowing methods can decrease it by calculating the accumulated **amount of density values of all hairs** that light goes through. On the other hand, aliasing caused by the problem that a computer cannot recognize too-thin hairs could be alleviated by image-space post-processing such as neighboring pixel averaging, though the quality depends strongly on the density of projected pixels.

Therefore, this thesis proposes an object-space method which realizes resolution-independent **anti-aliasing**. The method is an extension of the previous shadowing method that calculates the accumulated amount of density values of hairs, and **approximates a three-dimensional shape of hair** as a cloud of variable-size square splats with concentric opacity distribution. In this study, a novel evaluation method was also proposed to compare the extent of antialiasing in a quantitative manner. Comparison based on the novel evaluation method revealed that the proposed method outperforms the existing image-space anti-aliasing with minimum loss of computational speed.

#### Keywords

Hair; real-time rendering; anti-aliasing; object space; semi-transparent objects.

# 目 次

<b>1</b>	<b>序論</b>	<b>1</b>
1.1	研究背景 . . . . .	2
1.2	研究目的 . . . . .	2
1.3	本論文の構成 . . . . .	2
<b>2</b>	<b>関連研究</b>	<b>4</b>
2.1	音からアニメーションを自動生成する研究 . . . . .	5
2.2	アニメーションから音を生成する研究 . . . . .	5
2.2.1	サウンドレンダリング . . . . .	5
2.2.2	プロシージャルオーディオ . . . . .	6
2.3	音とアニメーションを同期させる研究 . . . . .	6
2.4	マーチングを行う演奏者の軌跡を解析する研究 . . . . .	7
2.5	本研究の新規性 . . . . .	7
<b>3</b>	<b>提案手法</b>	<b>8</b>
3.1	自動生成の流れ . . . . .	9
3.2	入力 . . . . .	9
3.3	出力 . . . . .	10
3.4	デバイス . . . . .	10
3.5	ソフトウェア . . . . .	10
3.6	3D モデル . . . . .	11
3.7	MIDI データから音情報への変換 . . . . .	11
3.8	音情報のモーションへの適用 . . . . .	12
3.8.1	指や腕、楽器のパーツへの適用 . . . . .	12
3.8.2	口元のメッシュへの適用 . . . . .	13
3.8.3	その他の部位への適用 . . . . .	13
3.9	提案手法の使用方法 . . . . .	14
<b>4</b>	<b>結果と評価</b>	<b>18</b>
4.1	実行環境 . . . . .	19
4.2	アニメーション生成結果 . . . . .	20
4.3	評価 . . . . .	21
4.3.1	実際の演奏シーンとの比較による評価 . . . . .	21
4.3.2	既存のアニメーションとの比較による評価 . . . . .	21
4.3.3	アンサンブルアニメーションの評価 . . . . .	21

4.3.4 システムの有用性の予想 . . . . .	21
<b>5 結論</b>	<b>25</b>
5.1 まとめ . . . . .	26
5.2 今後の課題 . . . . .	26
5.2.1 表情の豊かさの向上 . . . . .	26
5.2.2 モーションの種類の向上 . . . . .	26
5.2.3 モーションと音の関連付け . . . . .	27
5.2.4 楽器の種類および演奏者の増加 . . . . .	27
謝辞	28
公開文献	29
参考文献	30

# 図 目 次

1.1 演奏アニメーションが不自然であると感じることがあるかどうか . . . . .	3
2.1 複数の剛体がぶつかり破壊される様子 . . . . .	6
2.2 マーチングを行う演奏者の軌跡および誤差の表示例 . . . . .	7
3.1 音源から吹奏アニメーションを自動生成する流れ . . . . .	9
3.2 ウィンドシンセサイザ「EWI5000」 . . . . .	10
3.3 MIDI シーケンスソフトウェア「domino」 . . . . .	10
3.4 使用する 3D モデル . . . . .	15
3.5 番号付け . . . . .	16
3.6 音域による口元の変化 . . . . .	16
3.7 息継ぎをするときの口元 . . . . .	16
3.8 息継ぎ時のモーション . . . . .	17
4.1 Unreal Engine の初期設定 . . . . .	19
4.2 トランペット奏者 2 名の基本姿勢 . . . . .	20
4.3 トランペット奏者がピストンを押す様子 . . . . .	20
4.4 演奏者の口元 . . . . .	22
4.5 トランペット奏者 2 名とトロンボーン奏者 2 名の基本姿勢 . . . . .	23
4.6 トロンボーン奏者がピストンを動かす様子 . . . . .	23
4.7 曲の入りを合わせるために膝を使い楽器を下に向けた瞬間の様子 . . . . .	23
4.8 息継ぎしている瞬間の様子 . . . . .	24
4.9 トランペット奏者の指の動きは再現できているか . . . . .	24
4.10 演奏アニメーションが不自然であると感じことがあるかどうか . . . . .	24

# 表 目 次

3.1 MIDI データと音, 運指の対応 . . . . .	13
4.1 実行環境 . . . . .	19

---

## 第1章

---

### 序論

本章では、本研究の研究背景と研究目的について説明する。1.1節で研究背景、1.2節で研究目的、1.3節で本論文の構成について述べる。

## 1.1 研究背景

音楽演奏を題材としたアニメーションは多く存在する。テレビで放映されたアニメーションの例を挙げると、『のだめカンタービレ』、『けいおん！』、『響け！ユーフォニアム』が相当する。これらのアニメーションは、セル画であったり3DCGであったり、製作方法がさまざまであるが、いずれも実際に演奏する演奏者の身体の動きに近い演奏シーンが生成されている。楽器の輝きや形状なども忠実に再現されており、評判が高い。

一方、楽器を演奏するキャラクタの運指や身体の動きに注目すると、音楽と動きが完全に同期されていないことがあり、違和感を感じる。特に速いフレーズや、複雑なリズムを演奏するシーンで、このようなアーティファクトが起きやすい。また、表情からも不自然さを感じることがある。実際にアンケートをとったところ、吹奏楽やオーケストラの経験の有無に関わらず、不自然だと感じると回答した者が半数以上であった。

しかし、1つ1つの音と動きや表情を合わせるには、多くの時間と労力が必要となる。

## 1.2 研究目的

1.1節で述べた課題を解消するため、本研究では、管楽器を演奏するキャラクタの吹奏アニメーションを、音源から自動的に生成することを目指す。より具体的には、実際の演奏アニメーション制作フローに沿わせるため、楽曲は楽器（ウインドシンセサイザ）を用いて、MIDI（Musical Instrument Digital Interface）音源として生成する。次に、作成した楽曲を解析することにより、吹奏の情報を得る。最後に、得られた情報をキャラクタの身体の動きや表情、そして管楽器に適用することにより、音源に同期した自然な吹奏アニメーションを実現する。この手法の対象ユーザはアニメータとし、最終的にはアニメータのアニメーション制作時間と労力の削減を目的とする。また、本研究は、同研究室学士4年の武内と共同で行った。

## 1.3 本論文の構成

本論文は次章以降、以下のように構成される。

次章では関連研究を説明する。第3章では提案手法の仕組みを説明する。そして、第4章では自動生成結果を述べ、この結果に対する評価をまとめ。最後に第5章で結論を述べ、今後の課題に言及する。

なお、本研究の成果はVisual Computing/グラフィクスとCAD合同シンポジウム2017にてポスター発表<sup>[iii]</sup>を行った。そして、映像表現・芸術科学フォーラム2018（2018年3月）、Cyberworlds（2018年8月）において発表を行う予定である。

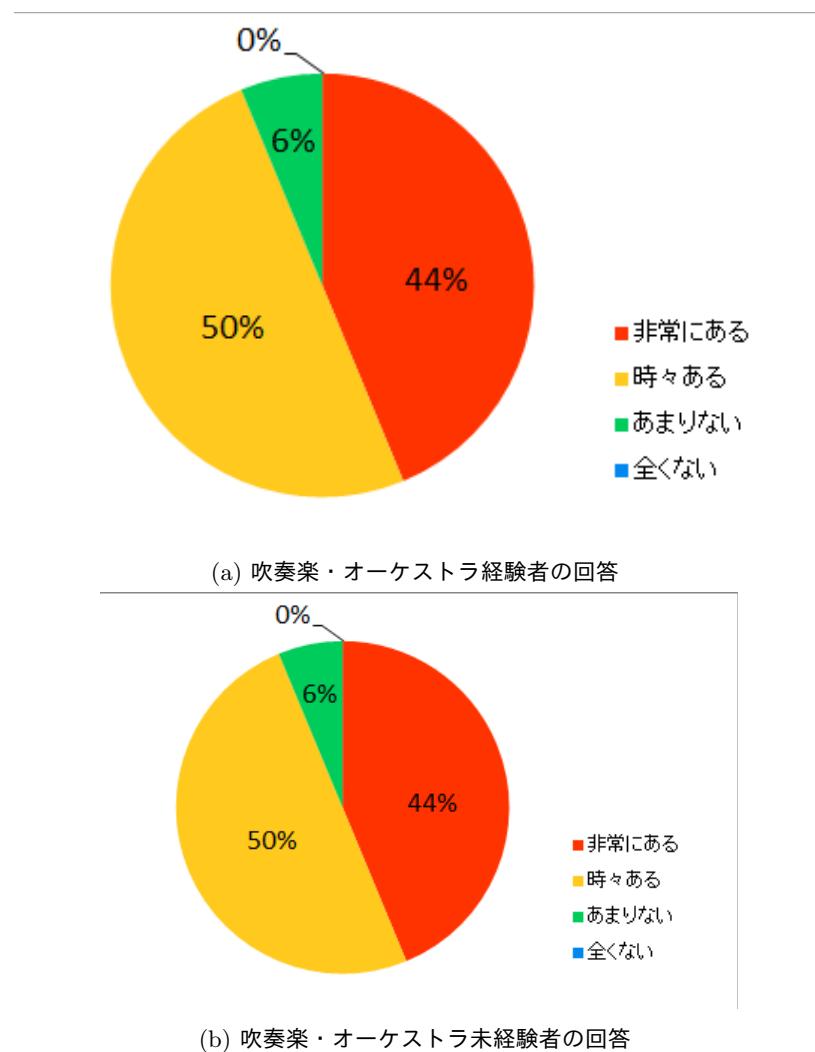


図 1.1 演奏アニメーションが不自然であると感じることがあるかどうか

---

## 第2章

---

### 関連研究

本章では 2.1 節で音からアニメーションを自動生成する研究を紹介し, ?? 節でアニメーションから音を自動生成する研究を紹介する. 2.3 節では音とアニメーションを同期させる研究, 2.4 節で本論文と同様, 吹奏楽に関連した研究を紹介し, 最後に 2.5 節にて本研究の新規性を述べる.

## 2.1 音からアニメーションを自動生成する研究

音とアニメーションを同期させることを目的として, 音からアニメーションを自動生成する研究は多く存在している.

物が落下するアニメーションを生成するには, 物の落下音と物が落下するタイミングを 1 つ 1 つ合わせる必要がある. Langlois ら [1] は, それらを正確に合わせるために, 物が落下する音が収録されている音源から, 物が落下するアニメーションを自動生成する手法を提案した.

キャラクタの口の動きと音声を合わせる, リップシンクも難しい課題となっている. キャラクタの口の動きに合わせて後から音声を録音するアフレコは比較的容易であるが, 音声から, その言葉を発している口元のアニメーションの生成は, 言葉と口の形, そして発するタイミングと口の動きを合わせる必要があり, ひじょうに難易度が高い. そこで, Edwards ら [2] は, 台詞が収録されている音源および台詞が記載されているテキストファイルを入力すると, その台詞を話している口元のアニメーションを自動生成する手法を提案した. 本手法では, 頬と唇だけを制御することにより, 自然な口元のアニメーションを自動生成している. 更に, 後から表情を編集することが可能となっているため, 最終的には自然な表情のアニメーションが完成する.

本論文の目的と同じように, 音源から楽器を演奏するアニメーションを自動生成する研究も多く行われている. Zhu ら [3] は, MIDI 音源からピアノを弾く指元のアニメーションを自動生成する手法を提案した. ピアノのような鍵盤楽器は, 指と音が 1 体 1 で対応していないため, フレーズにより指使いを変える必要がある. ある指が他の指の上や下を通るような指使いの時は, 指同士が衝突しないように考慮する必要もある. 彼らの手法では, これらの課題を解消することが可能である.

Yin[17] らは, Wave 音源からバイオリンを弾く手のアニメーションを自動生成する手法を提案した. 彼らは弦を押さえる左手の指元のアニメーションだけでなく, その指の動きが不自然に見えないような手首や腕の動きも再現している. 加えて, 弓を持つ右手の動きも自動生成している.

## 2.2 アニメーションから音を生成する研究

音とアニメーションを同期させることを目的として, 物理シミュレーションの結果を音に反映させる研究分野が存在する.

### 2.2.1 サウンドレンダリング

サウンドレンダリングという分野は, 物理アニメーションを生成すると同時に音を生成することにより, 双方を同期させることを目的とする分野である. CG-Arts の教育リポート, 『音を描き出す夢の実現』 [12] によると, 本分野は 1990 年代前半に登場し, 未だ実験的にとどまっている分野となっている. 2009 年には 1 つのシミュレーションパイプラインの中で, CG アニメーションと, そ

れに呼応した音を同時に生成するアルゴリズム??が考案された。本アルゴリズムは Zheng らにより考案され、水が流れる様子を再現したアニメーションと音の見事な同期が達成された。翌年には、同氏らにより破壊音を自動生成するアルゴリズム [5] が発表された。本アルゴリズムでは、さまざまな剛体の破壊音だけでなく、その衝撃が間接的におよぼす影響も考慮されているため、図 2.1 のような複数の物体がぶつかり、破壊されるような複雑なシーンでも、忠実に音が再現された。

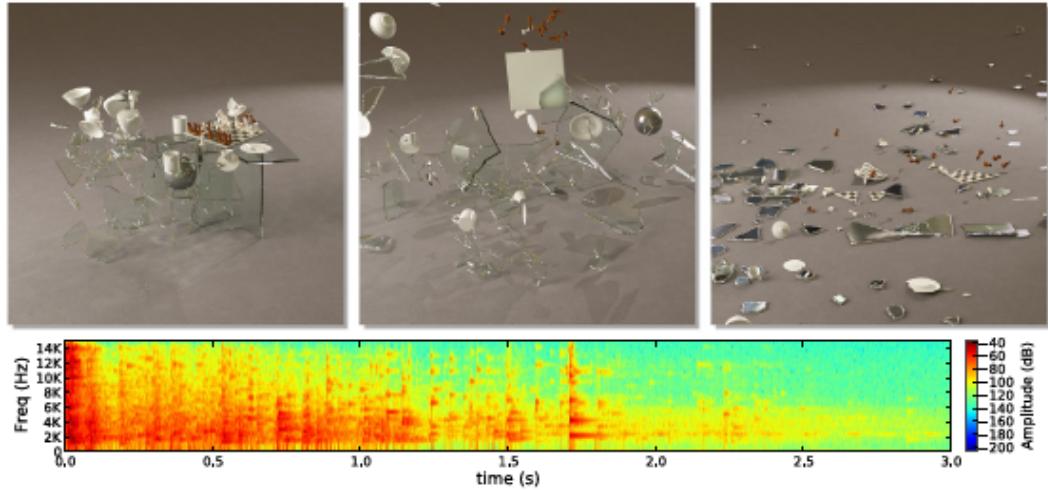


図 2.1 複数の剛体がぶつかり破壊される様子

### 2.2.2 プロシージャルオーディオ

アニメーションから音を自動生成する技術は、ゲームのような、使用可能なメモリが限られている場合には、できる限りメモリを節約したい。ゲームソフトを開発している株式会社スクウェア・エニックスは、メモリを節約するため、プロシージャルという技術を研究している。この技術は 2014 年の CEDEC[13] にて発表されており、あらかじめ作っておいた音をゲーム内で使用するのではなく、音をプログラムで作り出す技術をさす。ゲームでは、プレイヤの操作に合わせて音を鳴らすタイミングを内部で調節する必要がある。そのことからも、この技術はゲームにおいて特に必要な技術であると考えられる。

## 2.3 音とアニメーションを同期させる研究

片方からもう一方を自動生成する研究だけでなく、既に生成されている双方を編集することにより同期させる研究も存在する。Lee ら [6] は、MIDI 音源およびキャラクタのモーションデータを入力とし、双方を最小限編集することにより、MIDI 音源と同期したキャラクタのダンスアニメーションを生成する手法を提案した。

このような研究は、複数のカットを組み合わせて作成する、モンタージュの作成にも応用されている。Liao ら [7] は、BGM となる音源および複数のビデオクリップを入力することにより、BGM に合ったモンタージュを作成する手法を提案した。

## 2.4 マーチングを行う演奏者の軌跡を解析する研究

マーチングとは、演奏者が演奏しながら定められた経路に従って歩くことにより隊形を作る、吹奏楽の演奏形態である。本論文の筆者は、マーチングを行う1人の演奏者に注目し、その演奏者が歩くべき経路と実際の経路の誤差を表示するシステム図2.2を提案した。本システムにより、演奏者は自身の歩行の改善点を知ることができ、その結果マーチングの隊形の完成度を高めることが期待できる。なお、本研究の成果は NICOGRAPH2016 にてポスター発表[?]を行った。そして、一連の研究

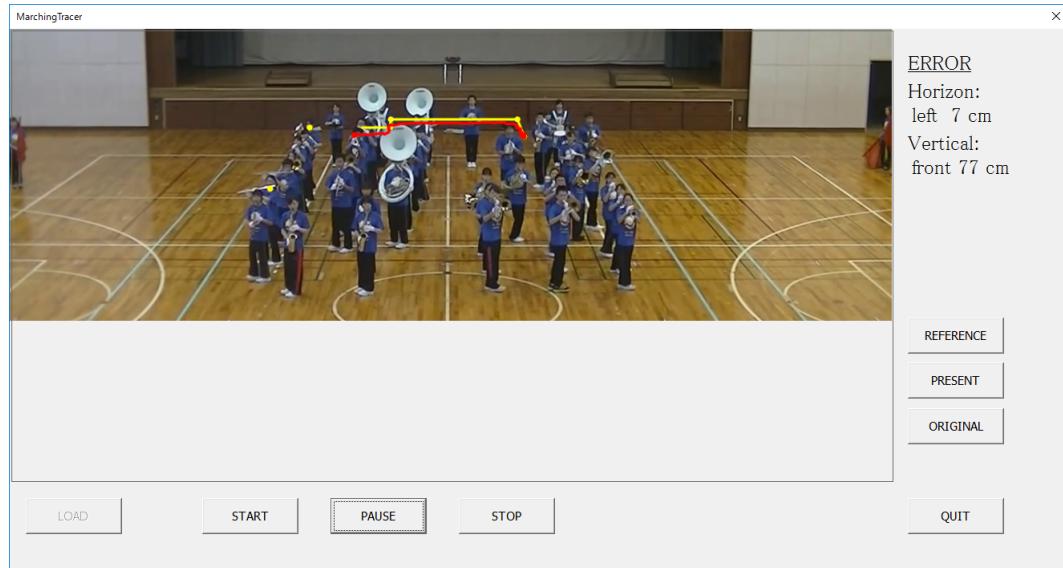


図 2.2 マーチングを行う演奏者の軌跡および誤差の表示例

をまとめた成果が画像電子学会誌の Vol.47[ii] に掲載される予定である。

## 2.5 本研究の新規性

アニメーションと音の同期や、吹奏楽をテーマとした研究は多く存在する。しかし、管楽器を演奏するキャラクタと音の同期に注目した研究は存在しない。また、2.1節で挙げた、ピアノやバイオリンを演奏するアニメーションの自動生成を行うための研究では、演奏者は1人となっている。したがって本研究の新規性は、管楽器を複数人で演奏するアニメーションを、音から自動生成するという点である。さらに、本論文ではトランペット奏者およびトロンボーン奏者に適用した例だけを示すが、音と指使いの対応表を用意することにより、すべての管楽器の演奏アニメーションが入力音源から自動生成が可能である。このように、あらゆる楽器への対応が可能である点も新規性である。

---

## 第3章

### 提案手法

---

本章では提案手法を詳しく説明する。まず、??節にて自動生成の大まかな流れを説明し、その後にそれぞれの工程について詳しく述べる。??節および??節にて入出力について説明し、??節にて使用するデバイス、??節にて使用するソフトウェア、??節にて使用する3Dモデルについて述べる。??節にてMIDIデータから音情報への変換方法、3.8節にて音情報のモーションへの適用方法を説明する。そして最後に、??節にて実際にシステムを使用する際の、使用方法について言及する。

### 3.1 自動生成の流れ

図3.1に、音源から吹奏アニメーションを自動生成する流れを示す。

実際のアニメーション制作フローに沿わせるため、音源の生成は楽器を用いて行う。次に、生成

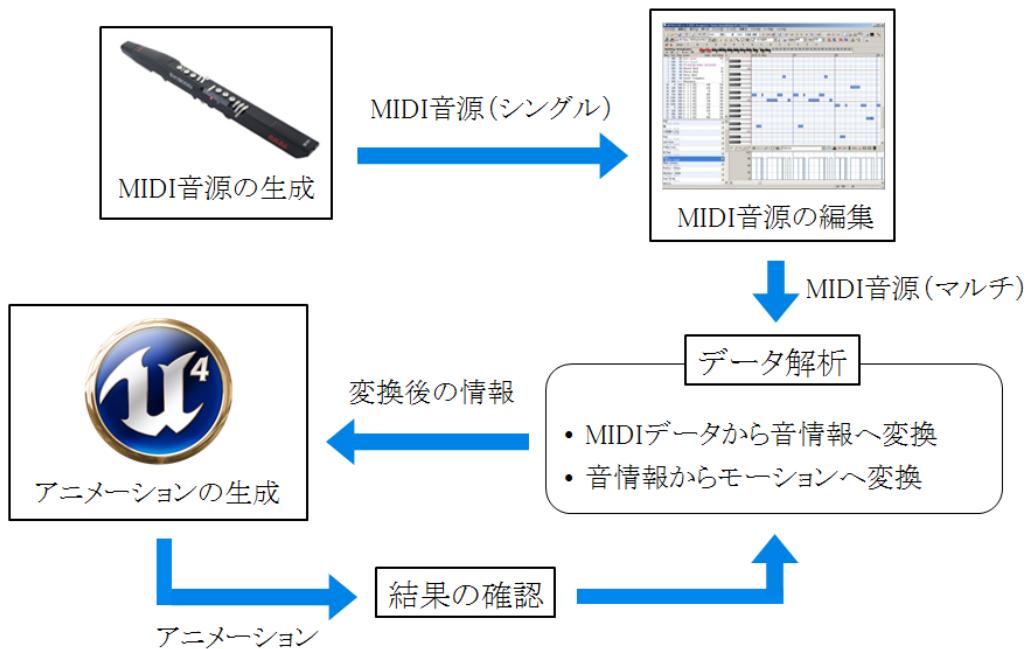


図 3.1 音源から吹奏アニメーションを自動生成する流れ

した音源を解析し、譜面データへ変換する。そして、アニメーション生成と同時に音源を流すことにより、音源に合わせてキャラクタが動くアニメーションが完成となる。

### 3.2 入力

入力する音源は、MIDI音源とする。ここで、MIDI音源は、MIDIという信号を受信して発音する音源のことである。よく使用されるmp3やwaveなどの形式とは異なり、中身が譜面データとなっているため、音の解析が比較的容易である。

このMIDI音源を生成する方法は、後述する。

### 3.3 出力

出力は、管楽器を演奏するキャラクタのアニメーションである。今回対象とする管楽器は、トランペット、トロンボーンである。この2本の楽器を選んだ理由は、吹奏楽ではとくに目立つ楽器であり、3Dモデルが入手しやすかったために選んだ。

### 3.4 デバイス

MIDI音源を生成するために、電子楽器であるウインドシンセサイザ「EWI5000」(図3.2)を使用する。このウインドシンセサイザは、さまざまな楽器の音を再現することが可能である。



図3.2 ウインドシンセサイザ「EWI5000」

### 3.5 ソフトウェア

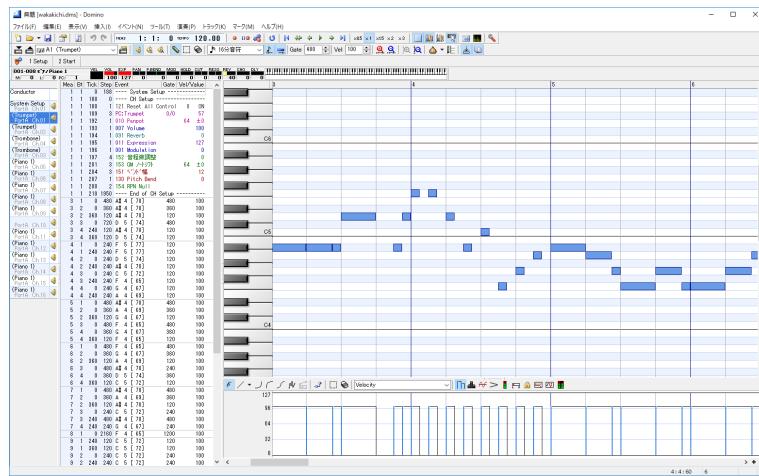


図3.3 MIDIシーケンスソフトウェア「domino」

3.4節で述べたウインドシンセサイザは、単音しか鳴らすことができないため、シングルチャネルの音源しか生成することができない。しかし、吹奏アニメーションを自動生成するためには、複数人で演奏しているマルチチャネルの音源が必要となる。そこで、ウインドシンセサイザで生成したMIDI音源を、フリーソフトウェアであるMIDIシーケンスソフトウェア「Domino」(図3.3)[8]に出

力し、重ねて何度も録音することにより、マルチチャンネルの音源を生成する。

また、アニメーションの生成には、Epic Games より開発されたゲームエンジン、Unreal Engine[9] を用いる。

### 3.6 3D モデル

使用する3Dモデルは、ユニティちゃん(図3.4(a)), トランペット(図3.4(b)), トロンボーン(図3.4(c))である。それぞれ、[14], [10], [11]からダウンロードした。なお、ユニティちゃんのライセンス条約は、webサイト[15]にて確認済みである。

ユニティちゃんは、主に以下の部位を制御する。

- 右指（トランペット演奏時）
- 右腕（トロンボーン演奏時）
- 背
- 腰
- 両足
- 口元

### 3.7 MIDIデータから音情報への変換

MIDIは、チャンクとよばれるデータブロックから構成され、先頭にヘッダチャンク、その後にトラックチャンクが続く。ヘッダチャンクには、チャンクタイプ、データ長さ、フォーマットタイプ、トラック数、タイムベース値という5つの値が格納されている。ここで、タイムベース値とは、四分音符1つが何クロックになるのかを表す値であり、一般的には48から960までの数字の中の、96の倍数が用いられやすい。一方トラックチャンクには、チャンクタイプ、データ長、トラックイベントデータが格納されている。提案手法では、これらの中からトラック数、タイムベース値、トラックイベントデータを楽譜データに変換した後に、アニメーションに適用できる形へとさらに変更する。

まず、音に関する情報を楽譜データに変換する方法について説明する。テンポ情報は、四分音符あたり何マイクロ秒なのか、という形で格納されている。音に関する情報は、トラックイベントデータとして以下のように16進数で格納されている。

9 0 4 8 6 4 8 1 7 0 8 0 4 8 0 0 8 3 6 0

この情報を以下のように2つずつペアにし、そこから音の種類と長さを取得する。

- 90: ノートオン。このタイミングで音を鳴らすことを意味する。
- 48: 音の種類（48はドを意味する。対応表は次項にて示す。）

- 64: 音の大きさ
- 81: 音を鳴らしている長さ（前半）
- 70: 音を鳴らしている長さ（後半）
- 90: ノートオフ。このタイミングで音を止めることを意味する。
- 48: 音の種類（48はドを意味する。）
- 00: 音の大きさ（音を止めているため、大きさはゼロとなる。）
- 83: 音を止めている長さ（前半）
- 60: 音を止めている長さ（後半）

音の長さは、以下のように10進数に変換する。ここでは、81 70を用いて説明する。

- まず、それぞれを2進数に変換する。81: 1000 0001, 70: 0111 0000
- そして、それぞれの最上位ビットを取り除いて合成し、10進数に直す。000 0001 111 0000 → 240

この例の場合、ドを240クロック分伸ばすことを意味する。以降、この値をデルタタイムとよぶ。デルタタイムを楽譜の情報に直すには、3.7節の冒頭で述べたタイムベース値を用いる。

最後に、このクロックを現実時間[s]に変換する。変換式は、(3.1)で表される。

$$\text{時間 } [s] = \frac{\text{デルタタイム } [\text{ticks}] \times 60[\text{seconds}/\text{minute}]}{\text{タイムベース値 } [\text{ticks}/\text{beat}] * \text{テンポ } [\text{beats}/\text{minute}]} \quad (3.1)$$

タイムベース値、テンポをそれぞれ仮に480, 120とすると、例で示した音情報は、譜面情報に直した結果『ドを0.25秒伸ばし、0.5秒休む』ことを意味すると分かる。

## 3.8 音情報のモーションへの適用

### 3.8.1 指や腕、楽器のパートへの適用

音を鳴らしている様子をアニメーションで再現するには、キャラクタの指や腕を介して楽器を制御する必要がある。トランペットはピストンの操作、トロンボーンはスライド管の操作により音を変えることができ、ピストンは3箇所、スライド管は止める場所が大きく分けて7箇所ある。トランペットのピストン番号を、吹き口に近い方から1-3(図3.5(a))、トロンボーンのスライドの位置を、吹き口に近い方から1-7(図3.5(b))と表すと、MIDIデータと音、運指の対応は表3.1となる。

??節で例として挙げた、『ドを0.25秒伸ばし、0.5秒休む』という譜形を演奏する場合、トランペットの場合は、1番と3番を0.25秒間押し続け、その後0.5秒間は休みのためそのまま、トロンボーンの場合は、スライドを3番まで移動させ、そのまま0.25秒間維持、その後0.5秒間は休みのためそのまま、という表現方法となる。

表 3.1 MIDI データと音、運指の対応

MIDI データ	音名（実音）	トランペット	トロンボーン
⋮	⋮	⋮	⋮
39	A (ラ, 442Hz)	2	2
3b	B (シ)	1・2・3	4
3c	C (ド)	1・3	3
3e	D (レ)	1・2	1
40	E (ミ)	2	2
41	F (ファ)	0	1
43	G (ソ)	1・2	2
45	A (ラ)	2	2
⋮	⋮	⋮	⋮

### 3.8.2 口元のメッシュへの適用

金管楽器を演奏する際、高音域の演奏時は、低音域に比べて口元が緊張する。緊張の程度には個人差があるが、本論文では、高音演奏時と低音演奏時の口元の違いを図3.6(??)のように、口の引き具合で表す。

休みが一定の時間以上続く場合、演奏者は息継ぎを行う。息継ぎのタイミングには個人差があり、演奏しているフレーズによっても異なるが、提案手法では息継ぎのタイミングを以下の3パターンに分ける。

- 休みが0.5拍以上～1拍未満：休みの間、ずっと息継ぎのモーションを行う。
- 休みが1拍以上～2拍未満：最後の1拍で息継ぎのモーションを行う。
- 休みが2拍以上：最後の2拍間で、息継ぎの予備モーションおよび息継ぎのモーションを行う。

息継ぎのモーションを行うときに、口元を図4.4bのように変形させる。その他のモーションについては次項で述べる。

### 3.8.3 その他の部位への適用

息継ぎをするときは口元だけでなく、上半身も動く。そこで、前項で述べたタイミングで、息継ぎのモーション図4.4b(b)、息継ぎの予備モーション図4.4b(c)を行う。なお、比較対象として通常の直立状態を図4.4b(a)に示すが、差が分かりづらい場合は、楽器と床の位置関係を見てほしい。

息継ぎ時以外にも演奏者は動く。例えば、任意のタイミングで重心を移動したり、テンポに合わせて身体や楽器を上下に揺らす。しかし実際、演奏時の動きには個人差があるため、モーションの大きさを調節するためのパラメタを指定することによる個人差の表現が可能である。現在はランダムでパラメタが設定される仕様となっている。そのため、複数人で演奏する際は、全員に異なるパラメタを

ランダムで割り当てることにより、個人差を表現する。以後、複数人で演奏することをアンサンブルとよぶ。アンサンブルには指揮者が存在しないため、曲の始まりは、リードを担当する演奏者が楽器や身体で合図をする。なお、重心移動や合図のモーションは、同研究室学士4年の武内が担当した。

### 3.9 提案手法の使用方法

提案手法を用いてアニメーションを自動生成する際は、以下の順序が必要となる。ここで、キャラクタや楽器のモデルは、あらかじめセッティングされているものとする。

1. MIDI 音源の作成
2. 音源データの相対パスをコードに記載し、コンパイル
3. Unreal Engine 4 を起動
4. 1. で生成した音源を BGM として設定
5. 再生
6. 結果の確認および修正

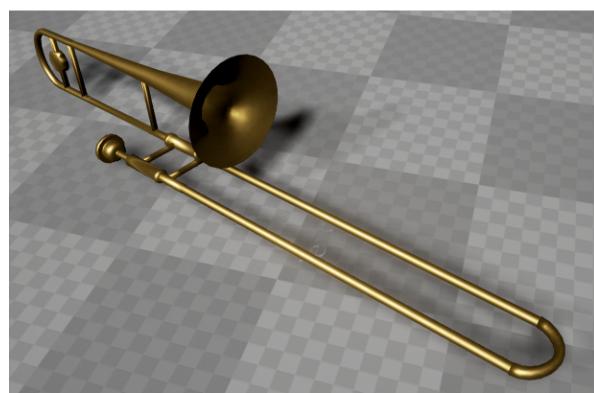
手順4.について、本来なら MIDI 音源を解析すると同時に音を流すべきであるが、現在の実装ではそれが不可能となっているため、解析する音源とは別に、流す音源として新たに設定する必要がある。また、このとき音のずれが生じる場合がある。その場合は少しの修正が必要となるが、最初の音だけを合わせることにより、それ以降はアニメーションと音が完全に一致する。



(a) ユニティちゃん



(b) トランペット



(c) トロンボーン

図 3.4 使用する 3D モデル

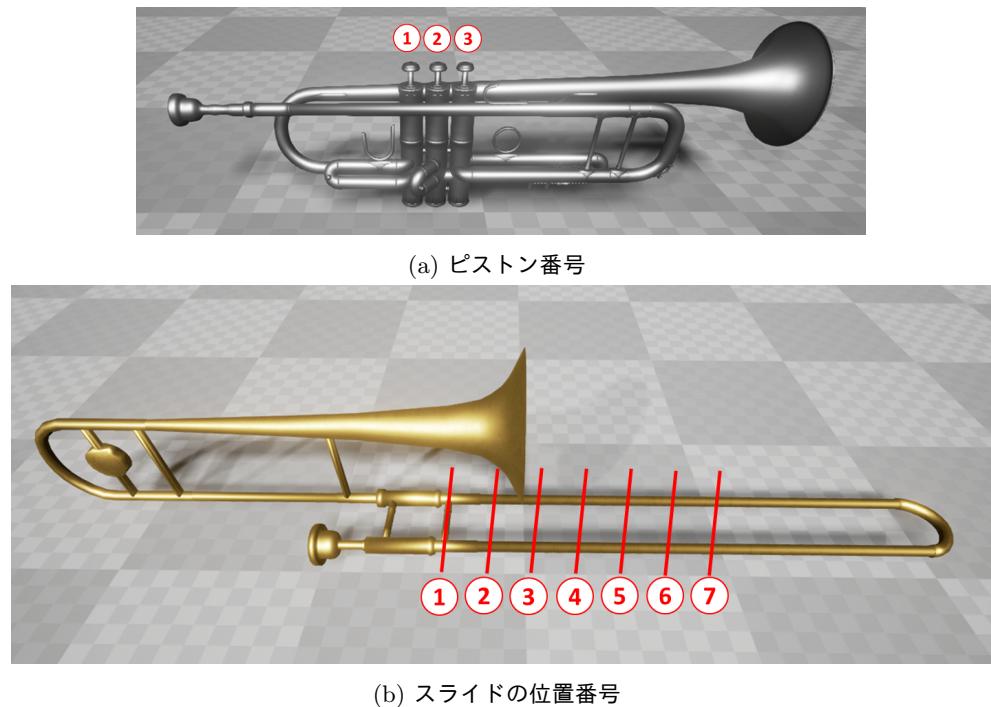


図 3.5 番号付け



図 3.6 音域による口元の変化



図 3.7 息継ぎをするときの口元



図 3.8 息継ぎ時のモーション

---

## 第4章

### 結果と評価

---

本章では、3章で紹介した手法によって自動生成した吹奏アニメーションについて記述する。4.1節では仕様したPCやシステムの仕様を述べ、4.2節では自動生成した吹奏アニメーションのキャプチャリング画像を示す。そして4.3節では自動生成結果を実際の演奏シーンや既存の吹奏アニメーションと比較することにより、提案手法を評価する。

## 4.1 実行環境

事前にUnreal Engineにて専用のプロジェクトを作成し、モーションのデータが記載されているUnreal Engine専用のファイルをインポートする。そして、キャラクタと、それぞれが演奏する楽器を図4.1のようにセッティングする。なお、最初にインポートするUnreal Engine専用のファイルには、キャラクタの姿勢データも存在するため、キャラクタの姿勢のセッティングは容易にできる。

実行環境は表4.1の通りである。

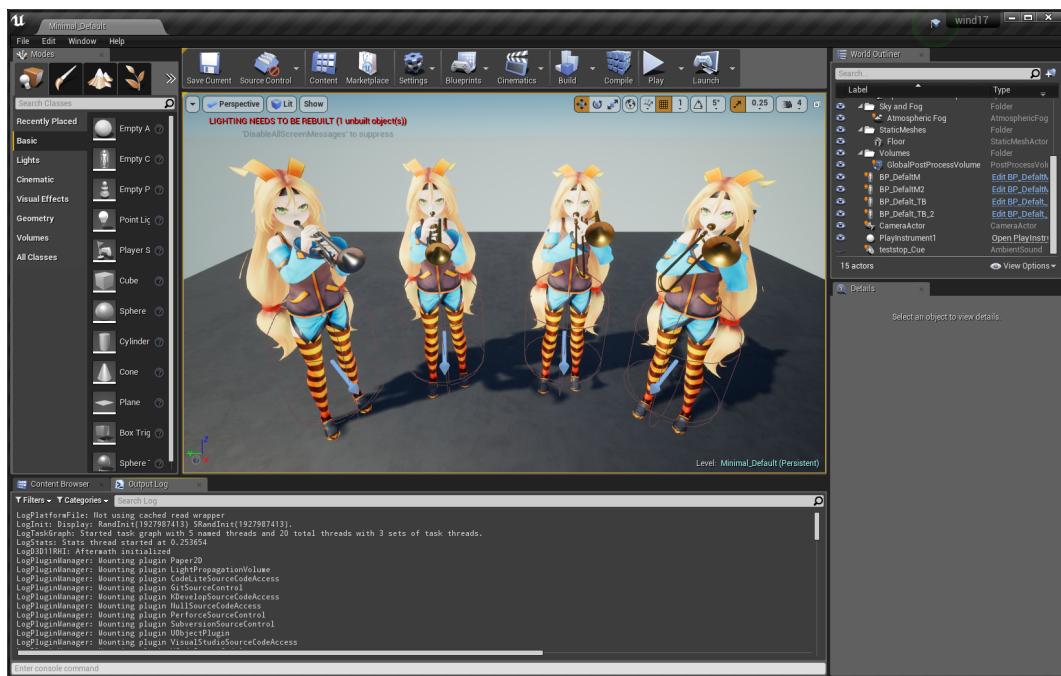


図 4.1 Unreal Engine の初期設定

表 4.1 実行環境

OS	Windows10 64bit
CPU	Intel®Core™i7-3930K
RAM	32.00GB
言語	c++

## 4.2 アニメーション生成結果

本研究で作成したアニメーションは、主に少人数編成であるアンサンブルアニメーションである。以下では、トランペット奏者2名で演奏するアニメーションのキャプチャリング画像と、トランペット奏者2名、トロンボーン奏者2名の計4名で演奏するアニメーションのキャプチャリング画像を示す。以下、それぞれについて制御している口元やボーンの制御について説明するが、前者のアニメーションは、トランペット奏者の口元や指元がズームインされているシーンであるため、主に口元や指元のボーンの制御について、後者のアニメーションは全体を俯瞰しているシーンであるため、前者のアニメーションにて述べなかったボーンの制御について記述する。

図??は、トランペット奏者2名の基本姿勢である。トランペット奏者は、図??のように指でトラ



図 4.2 トランペット奏者2名の基本姿勢

ンペットのピストンを押す。息継ぎをするタイミングでは、演奏者は図4.4(b)のように口を開ける。

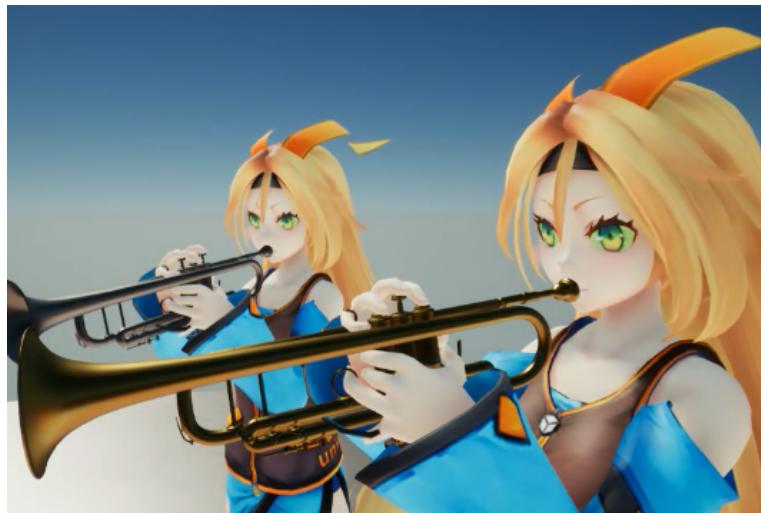


図 4.3 トランペット奏者がピストンを押す様子

また、高音を演奏するタイミングでは、図4.4(c)のように口元が緊張する。図??は、トランペット

奏者2名、トロンボーン奏者2名の基本姿勢である。トロンボーン奏者は、図??のように右手でスライドを動かす。曲の入りでは、タイミングを合わせるために、膝を使って楽器を縦に振る。図??は、膝を曲げ、楽器を下に向けた瞬間の様子である。息継ぎをするタイミングでは、演奏者は身体を反らす。図??のは、トランペット奏者2名が息継ぎをしている瞬間の様子である。

### 4.3 評価

自動生成した吹奏アニメーションに対して、以下の3つの評価を行った。

- 実際の演奏シーンとの比較による評価
- 既存のアニメーションとの比較による評価
- アンサンブルアニメーションの評価

以下、評価結果を、吹奏楽やオーケストラ経験者、吹奏楽やオーケストラ未経験者に分けて示す。

#### 4.3.1 実際の演奏シーンとの比較による評価

トランペット奏者、トロンボーン奏者が実際に演奏しているシーンと、自動生成した吹奏アニメーションを比較してもらい、以下の2項目についてアンケートをとった。

- 指や腕の動きは再現できているか
- 演奏している雰囲気はあるか

その結果を示す。

#### 4.3.2 既存のアニメーションとの比較による評価

#### 4.3.3 アンサンブルアニメーションの評価

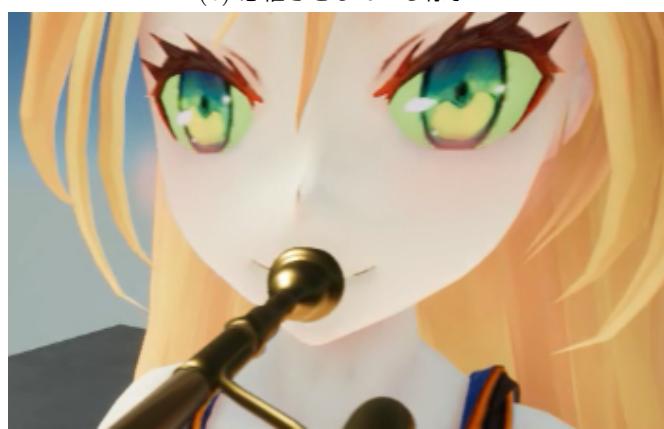
#### 4.3.4 システムの有用性の予想



(a) 通常の口元の様子



(b) 息継ぎをしている様子



(c) 高い音を演奏し口元が緊張している様子

図 4.4 演奏者の口元

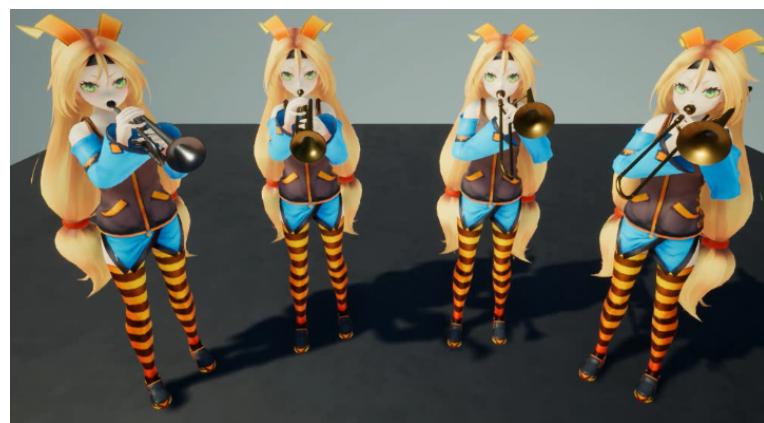


図 4.5 トランペット奏者 2 名とトロンボーン奏者 2 名の基本姿勢



図 4.6 トロンボーン奏者がピストンを動かす様子



図 4.7 曲の入りを合わせるために膝を使い楽器を下に向けた瞬間の様子



図 4.8 息継ぎしている瞬間の様子

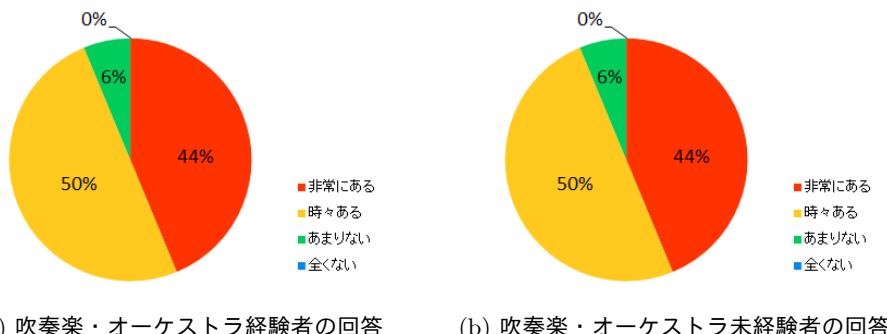


図 4.9 トランペット奏者の指の動きは再現できているか

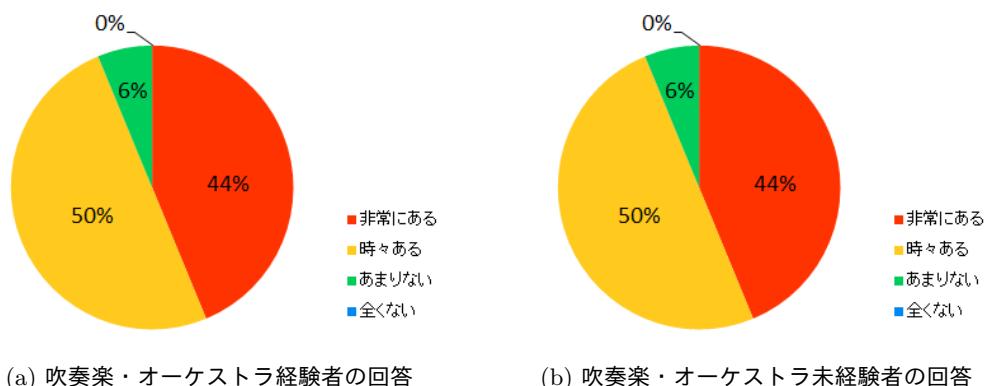


図 4.10 演奏アニメーションが不自然であると感じることがあるかどうか

---

## 第5章

---

### 結論

本章では本論文の結論を述べ、今後の課題に言及する。

## 5.1 まとめ

本論文では、音源から演奏アニメーションを自動生成することにより、音源に同期したアニメーションを自動生成する手法を提案した。音源は音情報を容易に解析できる MIDI 音源を使用し、そこから得た遠走の情報を Unreal Engine のキャラクタに適用することにより、運指や表情が音源に同期した吹奏アニメーションを少ない時間、少ない労力で生成することができた。特に音と 1 対 1 で対応する、トランペット奏者の指元や、トロンボーン奏者の腕の動きは、完全に音と同期した動きを再現できた。複数人で演奏するアニメーションでは、それぞれの動きのパラメタを 0 から 1 で割り当てるにより、基となるモーションは 1 つであったが、演奏者全員に異なる動きを適用できた。

## 5.2 今後の課題

今後の課題としては、表情やモーションの種類の向上、モーションと音の関連付け、楽器や演奏者の増加が考えられる。本節では各々について詳しく説明する。

### 5.2.1 表情の豊かさの向上

3.8.2 項で口元の制御について述べたが、演奏する際に動くのは口元だけではない。他の演奏者と目配せをすることや、音が長くて息継ぎできない場合、音が高い場合に辛そうな表情をすることがある。楽器に息を入れる際に頬が膨らむ演奏者もいる。これらの表情の変化もモデルに適用することにより、表情の豊かさが向上し、より自然なアニメーションが完成すると考えられる。方法としては、音の長さなどから判断できる情報については、音情報から自動的に算出、感情の変化のような、音情報から判断しづらい情報については、表情の遷移を記載するメタデータを用意することにより、対応できると考えられる。

しかし、目配せや辛い表情など、あらゆる表情の再現は、現状では Unreal Engine のみでは不可能である。口元を制御する際、Unreal Engine のモーフターゲットという機能を用いているが、この機能はブレンドシェイプが適用されていないメッシュには使用できない。ここで、ブレンドシェイプとは、あらかじめ用意された複数の表情モデルをパラメタの調整により組み合わせることで、さまざまな表情を作成するアニメーション手法をさす。今回用いたユニティちゃんは、デフォルトで口元にブレンドシェイプが設定されているため、モーフターゲットによる口元の制御は可能であるが、目配せや辛い表情などを再現するためのブレンドシェイプは、デフォルトの設定では不足している。そのため、あらゆる表情を再現するためには、他のソフトウェアを介してブレンドシェイプを設定する必要がある。

### 5.2.2 モーションの種類の向上

自動生成したアニメーションと、実際の演奏シーンと比較すると、キャラクタの動きに不自然な点が見つかる。その原因の 1 つとして、モーションの種類が少ないと挙げられる。楽器を演奏

する際の身体のモーションについては3.8.3項で触れたが、ここで述べたモーションだけでは足りない。例えば、音の高低に合わせて楽器を上下に揺らす演奏者や、円を描くように楽器を揺らす演奏者がいる。これらのモーションを追加し、モーションの種類を増やすことにより、さまざまな表現の実現が可能になると考えられる。

### 5.2.3 モーションと音の関連付け

自動生成したアニメーションと、実際の演奏シーンと比較したときに、キャラクタの動きが不自然に見える原因として他に考えられるのが、モーションと音の関連付けが完璧でないことが挙げられる。現在は、曲のテンポに従って身体が動く仕組みとなっており、モーションの種類や大きさは、ランダムに選択される仕様となっている。音の遷移とモーションを1対1で対応させたり、モーションを事前に指定するためのメタデータを用意することにより、モーションと音を関連付ける必要がある。

### 5.2.4 楽器の種類および演奏者の増加

本論文では吹奏楽の1つの演奏形態である、少人数で演奏するアンサンブルを想定し、楽器はトランペットとトロンボーンを選択したが、アンサンブルで使用される楽器はこの2本だけではない。また、将来的には吹奏楽のアニメーションの再現を目指す。そのため、楽器のモデルを増やす必要がある。それぞれの楽器に対応させるため、新たな実装を加える必要があるが、3.8.1項の表3.1に示した音と運指の対応表を作成するだけで、トランペット、トロンボーンと同じように吹奏アニメーションの再現が可能となる。また、今回は演奏者としてユニティちゃんを選択したが、演奏者それぞれの外見は異なっているべきである。演奏者の外見が異なると、キャラクタの体型に差が出る。体型に関わらず自然な吹奏アニメーションを生成するには、リターゲティングのための追加の実装や、動きの差を記述するメタデータが必要となる。

## 謝辞

本論文の執筆にあたり、ひじょうに多くのアドバイスを頂いたばかりでなく、私からの様々な相談もお聞きいただいた慶應義塾大学理工学部情報工学科の藤代一成教授に深く感謝いたします。私の至らなさから様々なご負担やご迷惑をおかけしたにも関わらず、辛抱強くご指導いただきありがとうございました。今後いかなる場で働く際にも、先生のご指導を思い出しながら仕事に励んでいきたいと思います。

本研究に取り組むにあたり、実装面において多くのアドバイスをいただいただけでなく、コンピュータグラフィクス分野における様々な課題に出会う機会をいただいた、現在は株式会社 DeNA に所属しておられる松岡徹氏、株式会社デジタル・フロンティア開発部の福田啓氏、斎藤弘氏、高山耕平氏並びに開発部の皆様に深く感謝いたします。

研究を進めるにあたり時には楽しく話し合い、時には真面目に議論を行い、共に助け合った研究室の皆様に深い感謝をいたします。皆様の存在が無ければ、私の研究はここまで進めることができませんでした。

研究をするにあたり、科研費基盤研究(B)25280037 及び慶應義塾大学博士課程教育リーディングプログラム オールラウンド型「超成熟社会発展のサイエンス」に多大なるご支援をいただいたことを感謝しております。

最後に、研究を進めるにあたりお世話になりました、すべての方に感謝を申し上げたいと思います。

## 公開文献

- [i] 堀井 絵里, 藤代 一成, 「マーチングバンドにおける演奏者個人を対象とした移動経路の計量可視化」, 芸術科学会 NICOGRAPH2016 予稿集, pp.119-120, 2016 年 11 月.
- [ii] 堀井 絵里, 藤代 一成, 「マーチングバンドの演奏者個人に注目した移動経路の計量可視化」, 画像電子学会誌, Vol. 47, 未決定, 2017 年 12 月.
- [iii] 堀井 絵里, 藤代 一成, 「音源に同期する運指に注目した吹奏アニメーションの自動生成」, Visual Computing/グラフィクスと CAD 合同シンポジウム 2017 DVD 予稿集, No. 41, 一橋講堂, 2017 年 6 月.

## 参考文献

- [1] T. R. Langlois and D. L. James, “Inverse-Foley Animation: Synchronizing Rigid-body Motions to Sound,” *ACM Transactions on Graphics*, Vol. 33, No. 4, Article No. 44, pp. 44:1-44:11, July 2014.
- [2] P. Edwards, C. Landreth, E. Fiume, and K. Singh, “JALI An Animator-Centric Viseme Model for Expressive Lip Synchronization,” *ACM Transactions on Graphics*, Vol. 35, No. 4, pp. 127:1-127:11, July 2016.
- [3] Y. Zhu, A. S. Ramakrishnan, B. Hamann, and M. Neff, “A System for Automatic Animation of Piano Performances,” in *Proceedings of Computer Animation and Virtual Worlds*, Vol. 24, No. 5, pp. 445-457, September 2012.
- [4] C. Zheng and D. L. James, “Harmonic Fluids,” *ACM Transactions on Graphics*, Vol. 28, No. 3, pp. 37:1-37:12, July 2009.
- [5] C. Zheng and D. L. James, “Rigid-body Fracture Sound with Precomputed Soundbanks,” *ACM Transactions on Graphics*, Vol. 29, No. 4, pp. 69:1-69:13, July 2010.
- [6] H. C. Lee and I. K. Lee, “Automatic Synchronization of Background Music and Motion in Computer Animation,” *Computer Graphics Forum*, Vol. 24, No. 3, pp. 353-361, September 2005.
- [7] Z. Liao, Y. Yu, B. Gong, and L. Cheng, “Audeosynth: Music-Driven Video Montage,” *ACM Transactions on Graphics*, Vol. 34, No. 4, pp. 68:1-68:10, July 2015.
- [8] TAKABO SOFT, “Domino,” 最終アクセス：2018年2月5日。[Online]<http://takabosoft.com/domino>
- [9] Epic Games, “Unreal Engine,” 最終アクセス：2018年2月5日。[Online]<https://www.unrealengine.com/ja/what-is-unreal-engine-4>
- [10] “Free3D,” 最終アクセス：2018年2月5日。[Online]<http://tf3dm.com/>
- [11] “3D EXPORT,” 最終アクセス：2018年2月5日。[Online]<https://jp.3dexport.com/>
- [12] CG-ARTS, “音を描き出す夢の実現,” 最終アクセス：2018年2月5日。[Online][https://www.cgarts.or.jp/report/rep\\_kr/rep0901.html](https://www.cgarts.or.jp/report/rep_kr/rep0901.html)
- [13] 4Gamer.net, “[CEDEC 2014] 物理シミュレーションの結果に合わせて音を生成する方法とは。スクエニの開発者が語るプロシージャルオーディオの応用例,” 最終アクセス：2018年2月5日。[Online]<http://www.4gamer.net/games/032/G003263/20140903036/>

- [14] unity-chan!, 最終アクセス : 2018 年 2 月 5 日. [Online]<http://unity-chan.com/>
- [15] unity-chan!, “ユニティちゃんライセンス条項,” 最終アクセス : 2018 年 2 月 5 日. [Online][http://unity-chan.com/contents/license\\_jp/](http://unity-chan.com/contents/license_jp/)
- [16] 井上 謙次, “Windows API による MIDI プログラミング,” 最終アクセス : 2018 年 2 月 5 日. [Online][http://www.deqnotes.net/midi/winapi\\_midiprog/winapi\\_midiprog.pdf](http://www.deqnotes.net/midi/winapi_midiprog/winapi_midiprog.pdf)
- [17] J. Yin, A. Dhanik, D. Hsu, and Y. Wang, “The Creation of a Music-driven Digital Violinist,” in *Proceedings of the 12th Annual ACM International Conference on Multimedia*, pp. 476-479, October 2014.