三次元構造を考慮した二次元オブジェクトのモデリングとアニメーション技法

北村, 真紀

https://doi.org/10.15017/1543987

出版情報：九州大学, 2015, 博士（芸術工学）, 課程博士
バージョン：

三次元構造を考慮した二次元オブジェクトのモデリングとアニメーション技法

Modeling and Animation of Pseudo-3D Objects

北村　真紀
Maki Kitamura

2015 年 9 月

概要

本論文はコンピュータグラフィックス (以下 CG) における疑似三次元モデルの三次元モデリングとそのアニメーションに関する一連の研究をまとめたものである。通常の三次元モデルは縦横に加えて奥行きの情報を持ち、二次元モデルは奥行きの情報を持っていない。しかしながら、二次元で表現された手描きアニメのキャラクタや二次元 CG を用いたゲームの地形の表現などは、奥行きを意識させることができる。本研究では、このように二次元の平面的なデータ構造でありながら、奥行き構造を表現しているモデルのことを「擬似三次元モデル」と呼ぶことにする。

本研究の目的は擬似三次元モデルを一般の三次元モデルと同様に操作、処理、動作させ、 取り扱うための方法の確立である。これによって次のような利点が考えられる。例えば、オブジェクトを任意の視点から見ることができるようになり表現の幅が広がることや、キャラクタアニメーション制作のコストが削減されることなどである。しかし、疑似三次元モデルは一般的な三次元モデルとは異なり独自の表現方法を持っているため、三次元モデルとして扱うために次の 2 つの課題を解決する必要がある。

1. 疑似三次元モデルという二次元のデータから、三次元モデルを作る際の奥行きを推測する手法の開発。
2. 生成された三次元モデルに動きを付ける際、疑似三次元モデルに適した動きを生成する手法の開発。

本論文ではこれらの課題を解決するために、疑似三次元モデルをシーンとキャラクタに分けて論じる。シーンとは建物や地形など広い範囲のもので、アニメーションを行わないオブジェクトを指す。キャラクタは作品の登場人物のようにアニメーションを行うオブジェクトのことを指す。

初めに疑似三次元モデルのシーンを三次元モデルとして扱う方法について論じる。本論文では疑似三次元モデルのシーンとして見下ろし型視点表示という表現手法に着目する。これは古くからコンピュータゲームで使われている表現手法で、二次元 CG である

が疑似的に三次元構造を表している表現手法である。見下ろし型視点表示では、真上からの情報と正面からの情報を同時に表現しているため、一般的な三次元 CG の表現とは見た目が異なる。そこで、本論文の手法では見下ろし型視点表示から三次元構造を生成する手法と、見下ろし型視点表示と三次元 CG の見た目を滑らかにつなぐ手法を考案し実装する。そして、見下ろし型視点表示という疑似三次元モデルを三次元モデルとして操作可能となることを示す。

次の章では、疑似三次元モデルのキャラクタを三次元モデルとして扱う方法について論じる。疑似三次元モデルのキャラクタとは漫画やアニメなどのキャラクタのように二次元の表現として扱われているが、三次元構造を知覚できるようなキャラクタのことを指す。本論文ではこの疑似三次元モデルのキャラクタを三次元モデルとして扱う手法について論じる。具体的には二次元で表現されたキャラクタのイラストを目や口などのパーツに分割し、それらのパーツを三次元のビルボードとして配置し、視点に応じてビルボードの形状を変形させることで、三次元モデルとしての操作を可能にする。これを行うために、本論文の手法では異なる視点から描かれた 2 枚のキャラクタのイラストとそれが描かれた方向を入力とする。初めにイラストを目や口などの閉領域に分割し、イラスト間で目と目、口と口といったようにそれぞれの閉領域を対応付ける。閉領域の対応付けには領域の類似度を計算し、類似度が最も高いものを対応付ける。閉領域が対応付けられた後、それらの閉領域の三次元の位置を推定し、その位置にビルボードとして閉領域を配置する。そして、各閉領域の輪郭線を特徴点ごとに対応付け、イラスト間での形状の補間を行うことで、視点が変わった時にビルボードの形状が変形し、見た目の変化を実現する。またこれらの手法を統合的に扱うことができるソフトウェアの開発も行う。

次の章では疑似三次元モデルを用いたキャラクタのためのアニメーション生成について論じる。漫画やアニメなどのキャラクタは誇張して表現されることがあるため、動きもまた誇張して表現される。従来の三次元 CG ではキャラクタに動きを付ける際に、実際の人間が動いたデータをモーションキャプチャで取得したものを使うことが多い。しかし、この動きを疑似三次元モデルのキャラクタにそのまま適用すると違和感が生じることがある。そこで本論文ではリミテッドアニメーションという手法に着目した。リミテッドアニメーションとは日本のアニメーション分野で用いられている表現で、動きを強調したり省略したりすることでメリハリのある表現を可能にする手法である。モーションキャプチャで得られたデータをリミテッドアニメーション風に変換することで、疑似三次元モデルのためのアニメーションの生成を実現する。

以上のように、本論文では疑似三次元モデルを三次元モデルとして扱う手法を確立するために、疑似三次元モデルである見下ろし型視点表示の三次元モデリング手法の開発、疑似三次元モデルのキャラクタのための三次元モデリングおよびアニメーション手法の

開発を行った。

目次

第 1 章　序論　8
1.1　はじめに 8
1.2　研究背景 9
1.3　本論文の構成 15

第 2 章　関連研究　17
2.1　シーンのモデリングに関する研究 17
2.2　手描きイラスト調のキャラクタを動かす研究 19
2.3　カートゥーン調のキャラクタのためのアニメーション表現 22

第 3 章　疑似三次元モデルを用いたシーンの三次元モデリング　24
3.1　課題と解決策 24
3.2　三次元モデリング手法 25
3.3　結果と考察 35

第 4 章　疑似三次元モデルを用いたキャラクタの三次元モデリング　37
4.1　課題と解決策 37
4.2　三次元モデリング手法 38
4.3　結果と考察 46

第 5 章　疑似三次元モデルのキャラクタのためのアニメーションの生成　49
5.1　はじめに 49
5.2　アルゴリズム 51
5.3　結果 55
5.4　考察 59

第 6 章　結論　60

6.1 本研究の成果 61
6.2 今後の展望と課題 62

参考文献 65

図目次

1.1 それぞれのモデル 9
1.2 CG 制作の流れ 10
1.3 三面図 11
1.4 カートゥーン調のキャラクタ 12
1.5 View-dependent geometry 12
1.6 リミテッドアニメーションにおける振り下ろし動作の強調 13
1.7 見下ろし型視点表示 14
1.8 疑似三次元モデルのキャラクタ 15
1.9 本論文の構成 16

2.1 自動生成された都市 17
2.2 Sketch Based Modeling 18
2.3 Sketch Based Modeling による地形の生成 19
2.4 二次元モデルに動きを付ける手法 20
2.5 Furusawa らの手法 20
2.6 Rivers らの手法 21
2.7 Yeh らの手法 22
2.8 アニメーションの際の見た目の強調 23

3.1 見下ろし型視点表示を用いたゲーム作品 25
3.2 モデリングインタフェース 26
3.3 タイルの種類とエッジの種類 26
3.4 基本地形の生成 27
3.5 高さの修正 28
3.6 階段状の形状の作成 29
3.7 凹形状と凸形状 30

3.8 茶色タイルにおけるエッジの有無での変化 30
3.9 青色のタイルを用いて生成される池の地形 31
3.10 赤色とベージュ色のタイルを組み合わせて作る家形状 32
3.11 家形状の作成 32
3.12 家の土台の作成 33
3.13 家の屋根の作成 33
3.14 見下ろし型視点表示と従来の三次元 CG での表示の違い 34
3.15 見下ろし型視点表示と三次元 CG での表示の滑らかな遷移 35
3.16 本手法により生成された結果 36
3.17 本手法により生成された結果にテクスチャマッピングを行ったもの . . . 36

4.1 入力データ 38
4.2 手法の概要 39
4.3 輪郭線部分の統合 40
4.4 類似度による対応付けの比較 43
4.5 三次元空間でのビルボードの位置 44
4.6 三次元空間での位置の修正 45
4.7 特徴点の対応付け 46
4.8 本手法により生成された画像の一部 48

5.1 コマ撮り 50
5.2 中無し 51
5.3 各フレーム間の SSD のグラフと姿勢 52
5.4 コマ撮りの実装 53
5.5 元のモーションデータ (a) と本手法 (b) とフレームを一度に抜く方法 (c) の比較 55
5.6 Pitch 57
5.7 Kick 57
5.8 Tennis 58
5.9 Baseball swing 58
5.10 モーションの一部の区間のキーフレーム 58

第 1 章

序論

1.1　はじめに

近年、コンピュータグラフィックス（以下、CG）技術の発達により、より写実的で迫力がある表現が可能となった。また、NPR(Non-Photorealistic Rendering) 分野の技術の発展により、従来の三次元モデルを用いた CG では表現が難しかった手描きイラストのような表現も可能になってきている。このような NPR の技術は、アニメーション制作やゲーム制作でも用いられるようになってきており、二次元モデルを用いて表現された手描きのキャラクタから三次元モデリングを行うことで、手描きの雰囲気を出しつつ三次元モデルとして自由に動かすことが可能となった。一般的に、キャラクタやオブジェクトを三次元モデルとして操作するためには、これらを三角形や四角形といった多角形で表現する三次元 CG モデリングを行うが、二次元モデルで表現されたキャラクタやオブジェクトは正確な三次元構造を考慮して描かれていないことが多く、三次元構造に無理が生じるような表現もある。例えば、漫画の登場人物で、正面から見た時の髪型と側面から見た時の髪型が三次元構造を考えるときに無理が生じるような場合がある。しかし、我々はそのようなキャラクタを見て違和感を覚えない。また、二次元 CG を用いたコンピュータゲームで用いられているシーンの表現でも、正確な三次元構造を考えると無理が生じるが、我々はその表現自体に対して違和感を感じないような表現がある。このように、三次元構造を考えるときに無理が生じるが、我々がそれを知覚するときには違和感を覚えないような表現を可能にするモデルを本論文では疑似三次元モデルと呼ぶ。疑似三次元モデルとは、基本構造を二次元とし擬似的に三次元構造を表すモデルである（図 1.1）。これは構造として二次元であるため、視点の変更や任意の動きの生成といった三次元モデルとしての操作はできない。しかし、三次元モデルとしての操作が可能となればより多くの表現が可能になる。そこで、本論文では疑似三次元モデルを三次元モデルと

して操作できるようにし、疑似三次元モデルのための動きの生成を目的とする。三次元モデルとして扱うためには、疑似三次元モデルの二次元データから三次元構造を定義し、三次元空間で動かす手法の開発が課題となる。本研究ではこの課題を解決するために、疑似三次元モデルをシーンとキャラクタに分けて論じる。シーンとは地形や建物の背景を指し、アニメーションをしないオブジェクトのことを指す。キャラクタは作品の登場人物や小物を指し、アニメーションをするオブジェクトを指す。また、手描きアニメーションのキャラクタのような疑似三次元モデルには従来の三次元ＣＧのアニメーション手法とは異なる独特のアニメーション手法があるため、それを実現する手法についても論じる。

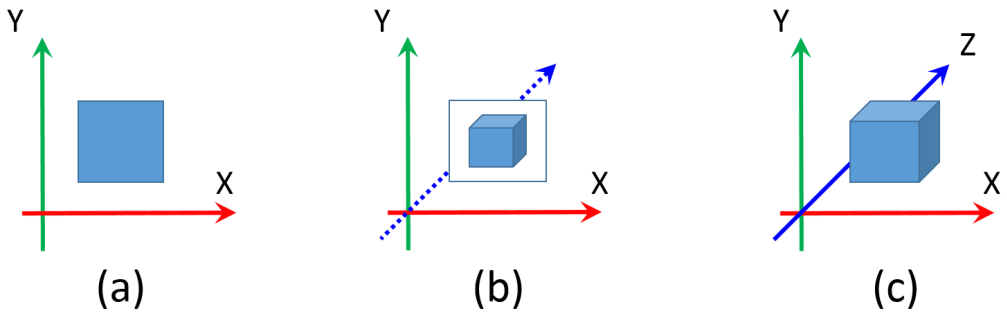


図 1.1　それぞれのモデル。(a) 二次元モデル。(b) 疑似三次元モデル。(c) 三次元モデル。二次元モデルは平面的な構造を表したものであり、三次元モデルは二次元モデルに奥行き情報を付加し立体的な構造を表したものである。疑似三次元モデルは基本構造を二次元とし疑似的に三次元構造を表すモデルである。このため、三次元構造は知覚できるが三次元空間での操作はできない。

1.2　研究背景

1.2.1　三次元 CG 制作の流れ

近年ではゲーム制作や映像制作において三次元モデルが使われることが多くなってきている。二次元モデルに比べ、三次元モデルでは視点の操作や、アニメーション作成にかかる時間のコストが少ないという利点がある。このため、何回も試行錯誤を行うことができ、作品の質の向上にかける時間を増やすことができる。一般的な三次元モデルの制作の流れは次のようになっている。まず初めに、アーティストが原案となるイラストを描き、原案をもとに CG デザイナーが三次元モデルを作成し、モーションデザイナーがキャラクタやオブジェクトの動きを作成し、レンダリングを行い結果を生成するといっ

た流れである [8](図 1.2)。CG デザイナーが三次元 CG モデリングを行う際には正面図、側面図、上面図という三面図（図 1.3）を使ってオブジェクトの情報を入力する必要があるため、原案でもそれらを考慮したデザインが必要となる。

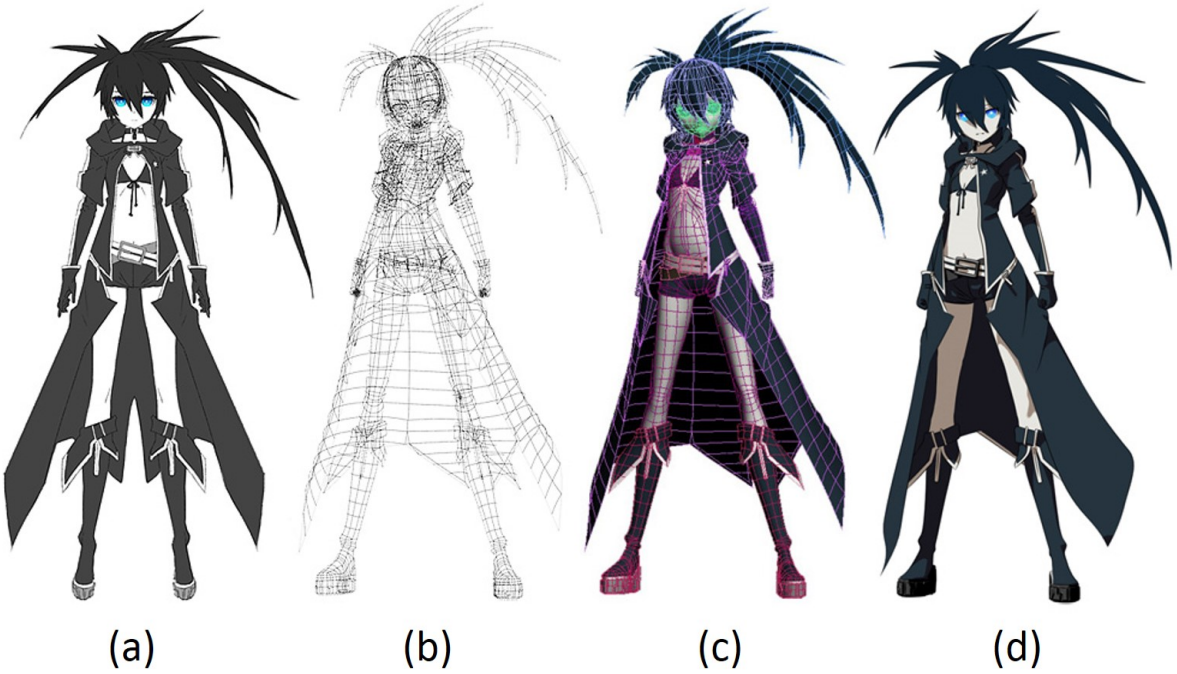


図 1.2　CG 制作の流れ。(a) 原案。(b) 三次元モデルのワイヤフレーム表示。(b) 三次元モデルに一般的なシェーディングを行ったもの。(d) セルシェーディングによりカートゥーン調のレンダリングを行ったもの。原案をもとに三次元モデリングを行い、ポーズを作成しレンダリングを行う。図は文献 [8] より引用。

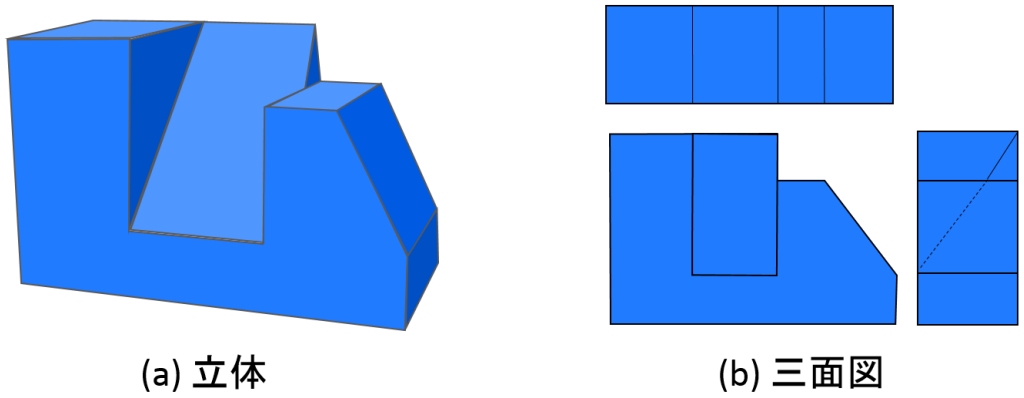


図 1.3　三面図。三次元モデリングを行う際には三面図を用いる。(a) の立体を三面図で表すと (b) のような表示になる。それぞれ真上、正面、真横から見た図が描かれている。

1.2.2　CG における非写実的な表現技法

コンピュータグラフィックスの研究分野において、光の反射や質感の表現といった物理特性を考慮した写実的な表現と共に、近年では絵画や手描き風の表現をするための Non-Photorealistic Rendering(NPR) という技術が発展してきた [49]。NPR の技術はカートゥーン調の表現ができるため、ゲームやアニメの CG でも用いられるようになってきている。カートゥーン調の表現の特徴としては、陰影を細かく付けずにベタ塗りのように少ない色数で表現していることと、キャラクタの輪郭線をはっきりと描くことが挙げられる (図 1.4)。これらの表現を CG で行うための研究は多く行われている [27, 41, 9]。三次元モデルでキャラクタを表現するためには、様々な方向からの見た目の情報を入力し三次元モデルを作る必要があるが、手描きのキャラクタを一つの三次元モデルで作成することは困難である。手描きのキャラクタの表現は誇張されることがあり、正確な三次元構造を考慮して描かれないことがあるからである。手描きの表現のように誇張された表現を三次元モデルで再現するために、図 1.5 のように視点によって三次元モデルの形状を変形させる View-dependent geometry [36] という研究が存在する。

図 1.4　カートゥーン調のキャラクタ。陰影を細かく付けずにベタ塗りではっきりとした陰影を付けている。輪郭線は黒色ではっきりと描かれている。

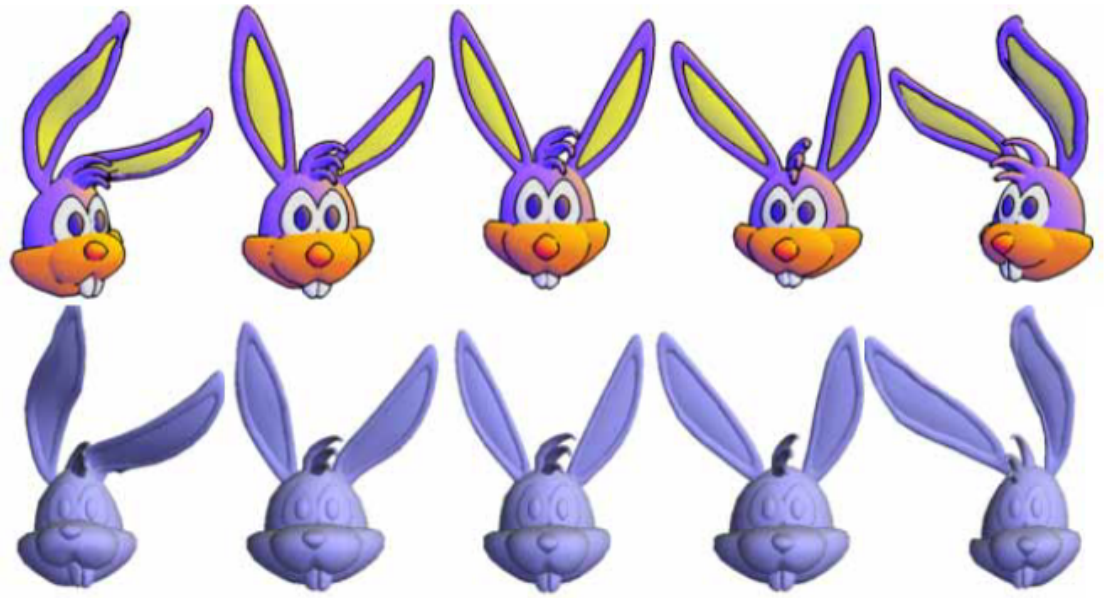


図 1.5　View-dependent geometry。視点によってキャラクタの形状を変形させることで常にキャラクタの耳がカメラを向くようにしている。図は文献 [36] から引用。

また、カートゥーン調のキャラクタの動きについても独自の表現方法がある。写実的な CG 表現での動きの生成は物理モデルやモーションキャプチャで実際に人が作った動きで表現されることが多いが、カートゥーン調のキャラクタにそのような動きを適用すると違和感が生じることがある。カートゥーンアニメではキャラクタの見た目だけでなく、動きも誇張されるからである。例えば、剣の振り下ろし動作のように勢いがある動きに関しては速さを強調するために、あえて動作の一部を抜いて表現することがある（図 1.6）。また、人間の動作では実際には細かい振動があるが、カートゥーンアニメではそのような細かい動きはあえて省略することで、動作をわかりやすく表現するということもある。このような手法を用いたアニメーションはリミテッドアニメーションと呼ばれている。

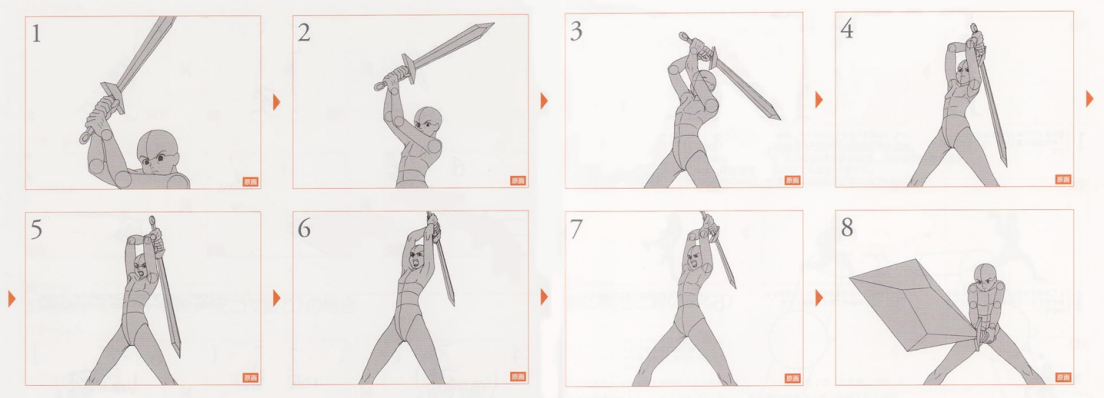


図 1.6　リミテッドアニメーションにおける振り下ろし動作の強調。キャラクタの動きを強調するために動作の一部を抜いている。ここでは、7 コマ目と 8 コマ目の間の動きを描かないことで剣の振り下ろし動作の速さを強調している。図は文献 [50] から引用。

1.2.3　二次元モデルと三次元モデルと疑似三次元モデル

CG には平面的な要素を表現する二次元モデルと二次元モデルに奥行き情報を付加することで立体構造を表現する三次元モデルがある。三次元モデルは立体構造を持っているため、様々な視点からの見た目を表現でき、より豊かな表現が可能となる。一方で、三次元モデルで表現するためには、表現したい物体の正面や側面など様々な方向から見た情報を定義する三次元モデリングという作業が必要であり、これは二次元モデルを扱うことに比べ非常に手間がかかってしまう。一般的に、三次元モデリングを行うためには最初に二次元で原案となるイラストを描きそれをもとにモデリングを行うが、漫画やアニ

メなどのキャラクタは誇張して表現されることがあるため、正面からの見た目と側面からの見た目に無理が生じ、三次元モデリングが困難になる場合がある。しかし、我々はこのようなキャラクタが動いている様子を見ても形状に関して違和感を覚えることはない。

このように、三次元構造を考えるときに無理が生じるが、我々がそれを知覚するときには違和感を覚えないような表現があり、疑似三次元モデルではそのようなものを表現できる。本研究では疑似三次元モデルをシーンとキャラクタに分けて論じる。疑似三次元モデルのシーンとして見下ろし型視点表示という表現方法がある（図 1.7）。これは、コンピュータゲームで用いられている表現であり、ゲームの舞台となる建物や地形を表現するものである。この表現において、水平方向の情報は幅を表しており、垂直方向の情報は高さあるいは奥行きを表している。見下ろし型視点表では真上からの情報と正面からの情報が同時に表現されているため、従来の三次元 CG では表現できない。しかし、我々は見下ろし型視点表示で表現された立体構造を違和感なく知覚できる。

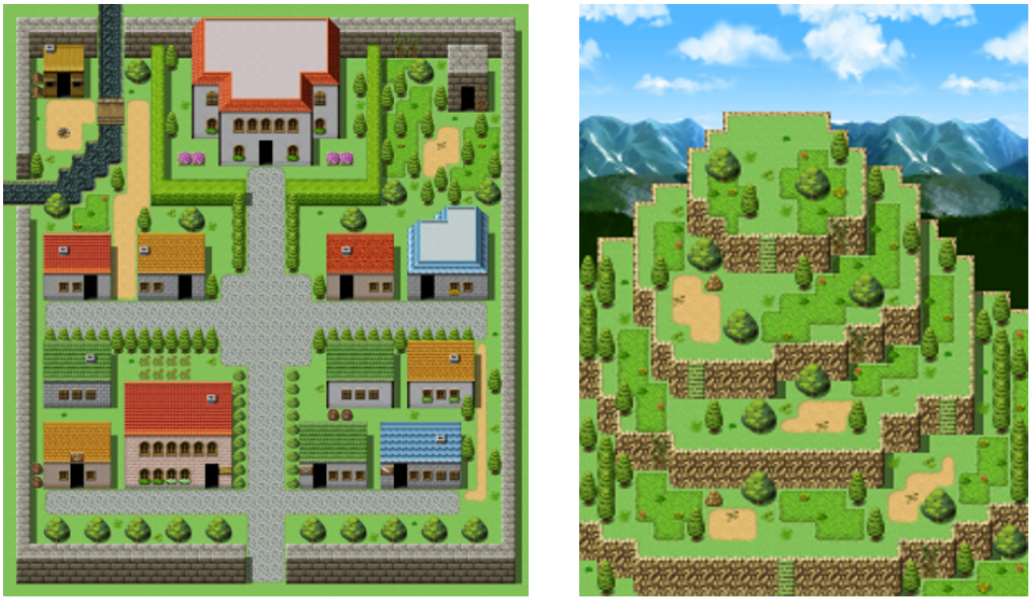


図 1.7　見下ろし型視点表示。正面からの見た目と真上からの見た目が混在している。画像は文献 [5] から引用。©2011 KADOKAWA CORPORATION./YOJI OJIMA。

疑似三次元モデルのキャラクタとしては、手描きの漫画やカートゥーンアニメのキャラクタが挙げられる (図 1.8)。手描きのキャラクタは誇張して表現されることがあるため、正面からの見た目と側面からの見た目に無理が生じることがある。しかし、我々が作品を鑑賞する上ではそのキャラクタの形状に違和感を覚えない。

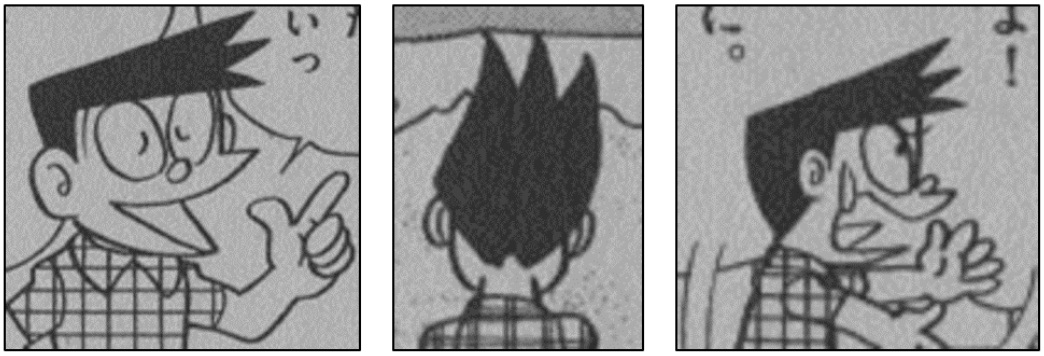


図 1.8　疑似三次元モデルのキャラクタ。このキャラクタでは正面からの見た目と横や後ろからの見た目から三次元モデルを作成することは困難である。 図は文献 [51] から引用。©藤子プロ 2014。

疑似三次元モデルは三次元構造を知覚することはできるが、奥行き情報を持っていないためそのまま三次元モデルとして扱うことはできない。しかし、三次元モデルとしての操作が可能になれば、視点の変更や姿勢の変更が可能になり、より豊かな表現が可能となる。本研究では、疑似三次元モデルを三次元モデルとして扱うことを可能にするための手法の開発を行う。

1.3　本論文の構成

疑似三次元モデルは基本構造は二次元であるため、そのまま三次元モデルとして扱うことはできないが、三次元モデルとして扱うことができれば任意の視点での見た目の作成や、キャラクタのアニメーション作成が可能となり表現の幅が広がる。そこで本論文では疑似三次元モデルを三次元モデルとして扱うための手法を開発するために、次の 2 つの課題を解決する。

1. 疑似三次元モデルという二次元のデータから、三次元モデルを作る際の奥行きを推測する手法の開発。
2. 生成された三次元モデルに動きを付ける際、疑似三次元モデルに適した動きを生

成する手法の開発。

1 つ目の課題に関しては、アニメーションを行わない疑似三次元モデルのシーンと、アニメーションを行う疑似三次元モデルのキャラクタに分け、それぞれを三次元モデルとして扱うための手法の考案を行う。2 つ目の課題に関してはアニメーションを行うキャラクタに焦点を当て、疑似三次元モデルに適した動きの生成を行う。本論文の構成は次の通りである（図 1.9)。第 2 章ではシーンやキャラクタのモデリングについての既存研究とキャラクタアニメーションに関する既存研究について論じる。第 3 章では建物や地形を表す疑似三次元モデルのシーンを三次元モデルとして扱う手法について論じ、第 4 章では作品の登場人物を表す疑似三次元モデルのキャラクタを三次元モデルとして扱う方法について論じる。第 5 章では疑似三次元モデルのキャラクタのためのアニメーション表現について論じる。そして、第 6 章で本論文の総括について述べる。

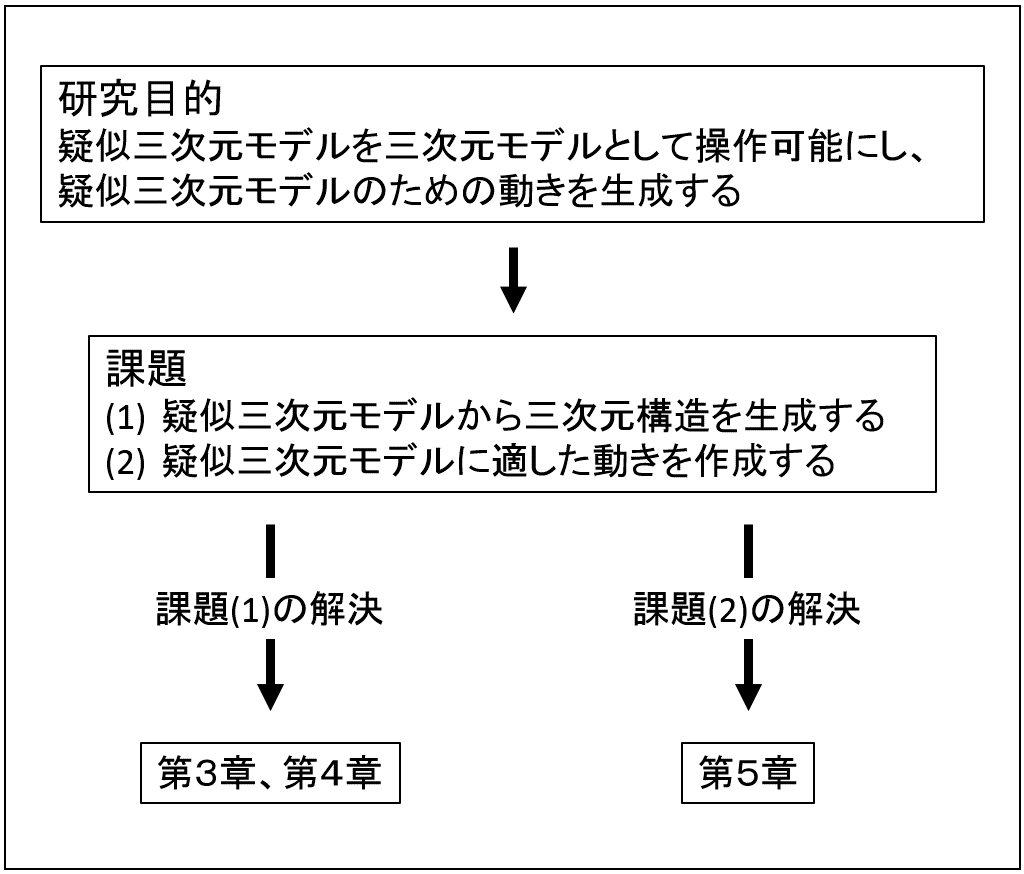


図 1.9　本論文の構成。

第 2 章

関連研究

2.1　シーンのモデリングに関する研究

建物や地形といったシーンを CG で扱うための研究は古くから多く行われている。映像制作やゲーム制作などで用いるための都市や地形を CG で表現するためには膨大な量の建物や地形のモデリングを行う必要がある。この作業を軽減するため、ビルや道路などの生成規則を作りその生成規則に従って自動的に都市や地形を生成する研究が発展してきた [35, 42](図 2.1)。この研究分野は成熟し、研究成果は実用化され映像制作の現場でも用いられている [2]。

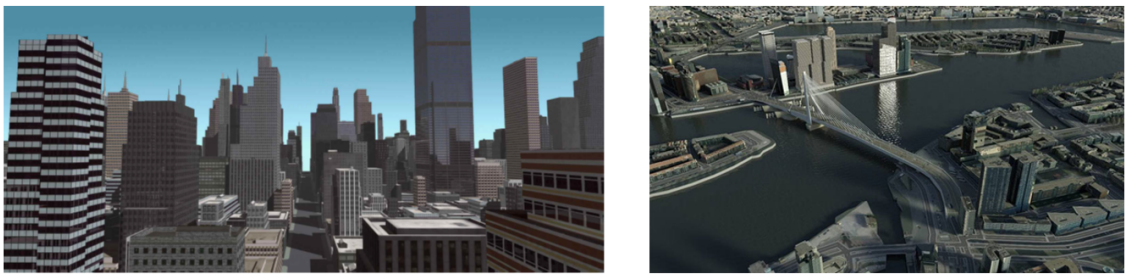


図 2.1　自動生成された都市。画像は文献 [35]（左）、文献 [2]（右）より引用。

実際の写真から建物や地形の三次元モデルを生成する研究も多く行われており、この研究は Image Based Modeling の分野で発達してきた。複数枚の写真から立体を生成する研究や [14, 47]、1 枚の写真のみで立体を生成する研究 [20, 33, 24] がある。これらの研究は写実的な CG 表現での表現の幅を広げ、制作コストの削減に貢献してきた。しかし、これらの手法は写実的で本物の地形を再現しているため、非写実的で誇張された表現

の三次元モデリングは困難である。

手描きのイラストのように非写実的な二次元の表現から三次元モデルを生成する研究は Sketch Based Modeling（SBM）という研究分野で発展してきた。SBM の技術を用いることで、ユーザは絵を描くような簡単なストロークで三次元モデリングができるようになり、三次元モデリングに熟練していない初心者のユーザでも直観的に三次元モデリングができるようになった (図 2.2)。SBM においてユーザの入力は二次元の手描きのストロークであるため、ユーザ入力から三次元モデルを作るためには奥行きの情報を付加する必要がある。奥行きの情報を付加するためには追加のユーザストロークを必要としたり、奥行き情報を計算する規則を用いたりする [22, 18, 25]。

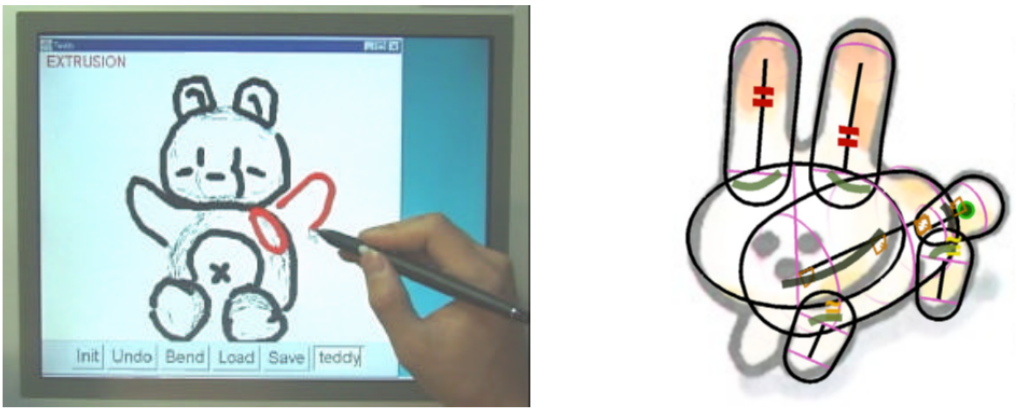


図 2.2　Sketch Based Modeling。手描きのストローク入力で三次元モデルを作成することができる。画像は文献 [22]（左）、文献 [18]（右）より引用。

SBM の分野でもシーンのモデリング技術が開発され、手描きのストロークで任意の地形形状が生成できるようになった [44](図 2.3)。任意の形状を作ることができる一方で、疑似三次元モデルからの三次元モデリングには困難が生じることがある。SBM では任意のユーザストロークから三次元形状を作ることができるとはいえ、三次元モデルを作成するためには正面や側面などの様々な視点からの見た目は一致しなくてはならず、見下ろし型視点表示のように従来の三次元 CG では表現できないような地形のモデリングはできない。

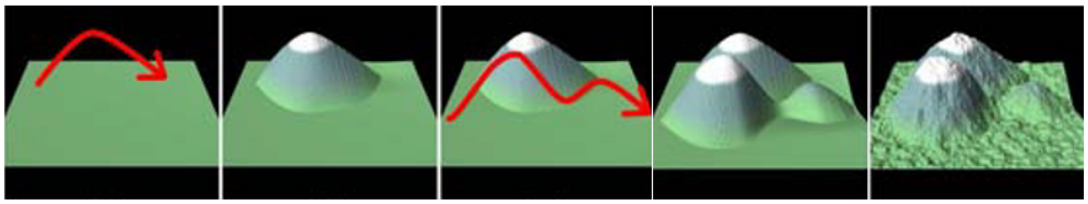


図 2.3　Sketch Based Modeling による地形の生成。三次元の地形に二次元のストロークを描くことで直観的に地形の作成ができる。画像は文献 [44] より引用。

2.2　手描きイラスト調のキャラクタを動かす研究

三次元 CG において、手描きのイラストのような表現を行うための研究は NPR の分野で発展してきた [15, 38]。しかし、手描きのキャラクタのイラストにおいては正面から見た状態と側面から見た状態が一致しないことがあり、三次元モデルを作ること自体が困難な場合もある。このような問題に取り組んだ研究が View-dependent geometry である [36]。この手法はキーとなる視点ごとに三次元モデルを歪め、線形補間を行うことで、静止した物体として見ると矛盾が生じる形状の表現を可能にした。一方で、この手法は視点ごとに参照用のイラストを用意し、三次元モデリングを行う必要がある。

また、二次元モデルのキャラクタアニメーションのための研究も多く行われている (図 2.4)。Igarashi らは 1 枚の入力画像をメッシュ分割し、形状の特徴を保ったままメッシュを変形することでキャラクタのアニメーションを可能にした [23]。また、1 枚のキャラクタ画像にスケルトン構造を考慮してモーションキャプチャの動きをつける研究もある [19, 34]。複数枚の画像を用いてキャラクタに動きをつける手法としてモーフィングの手法がある [11, 40, 46]。Baxter らはカートゥーン調の手描きのイラストをモーフィングを用いて動かす研究を行った [10]。これらの手法は固定された視点でのキャラクタの見た目での表現は可能であるが、視点の変更により変形の途中で遮蔽を伴う部分が生じるような表現には対応できない。これは遮蔽された部分が欠損して補間時に対応が取れないためである。

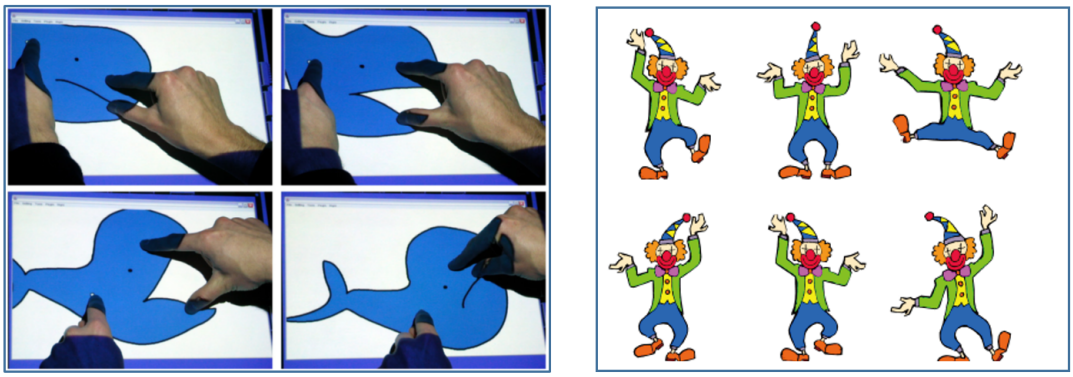


図 2.4　二次元モデルに動きを付ける手法。これらの手法では平面的な動きは作成できるが、振り向き動作のように変形の途中で遮蔽が生じるような動きを作ることはできない。画像は文献 [23]（左）、文献 [10] より引用。

遮蔽された領域があっても補間が可能な手法として Whited らは BetweenIT を提案した [45]。これは 2 つのキーフレームとなる線画のストロークを対応付け、対数螺旋を用いてストロークの補間を行うものである。この手法では遮蔽が生じる部分にユーザがストロークを追加することで遮蔽部分の補完を行うことができるが、奥行きを考慮せずストロークの対応付けのみで画像の補間を行うため、視点が大きく変わった時の見た目の変化に対応できない。Furusawa らは顔の線画の正面図と側面図に対し、特徴点を対応付けて補間を行った [17]（図 2.5）。彼らの手法でも遮蔽された線は扱うことができるが、 奥行きを考慮しておらず、領域の一部だけが遮蔽される様子を表現できない。

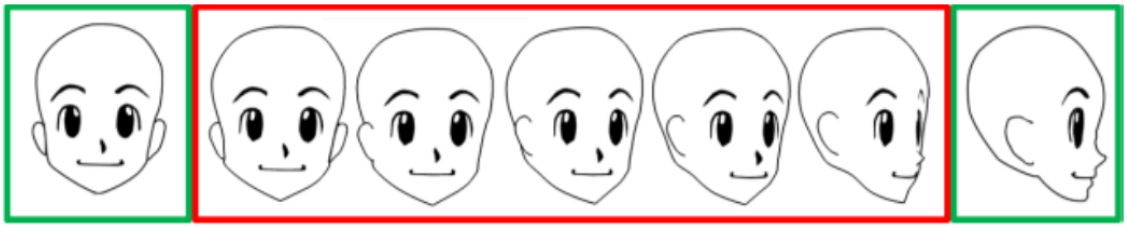


図 2.5　Furusawa らの手法。緑枠が入力画像で赤枠が出力結果。この手法では遮蔽された領域があっても補間が可能である。画像は文献 [17] より引用。

疑似三次元モデルの中でも、様々な視点からの見た目が一致しないような形状を扱う手法として、視点に応じて二次元モデルを変形させることで三次元 CG のような見た目を表現する手法がある。Di Fiore らは複数視点から見た手描きのイラストの補間を行った [16]。この手法ではイラストの各パーツの奥行きを手動で設定する必要がある。本論文の手法では、各視点での視線方向のベクトルを使うことで奥行きを自動で推定で

きる。Rivers らは疑似三次元モデルから三次元モデルを作成するインタフェースを提案した [37]。これはユーザが様々な視点から見たキャラクタの目や口などのパーツをストロークで描き、パーツの形状の補間を行うことで様々な視点からの見た目を表現できるシステムである（図 2.6）。この手法では複数視点から見たストロークの補間を行うことはできるが、ストロークをユーザがひとつずつ作成することは非常に時間がかかる。本論文の手法ではアーティストがキャラクタデザインの際に描いた正面や側面から見たイラストを流用できるため、ストローク作成の手間がかからない。Yeh ら [48] はイラストの表と裏の面を用いて、疑似的に三次元を表現する手法を提案した (図 2.7)。この手法では球や回転体のように回転時に輪郭の形状が変わらない物体であれば遮蔽を伴う表現を行うことができるが、回転時に輪郭の形状が変化するような物体の表現はできない。本論文の手法では複数の視点から見た手描きのイラストを用いて形状を補間するため、輪郭の形状が変化するような場合にも対応できる。

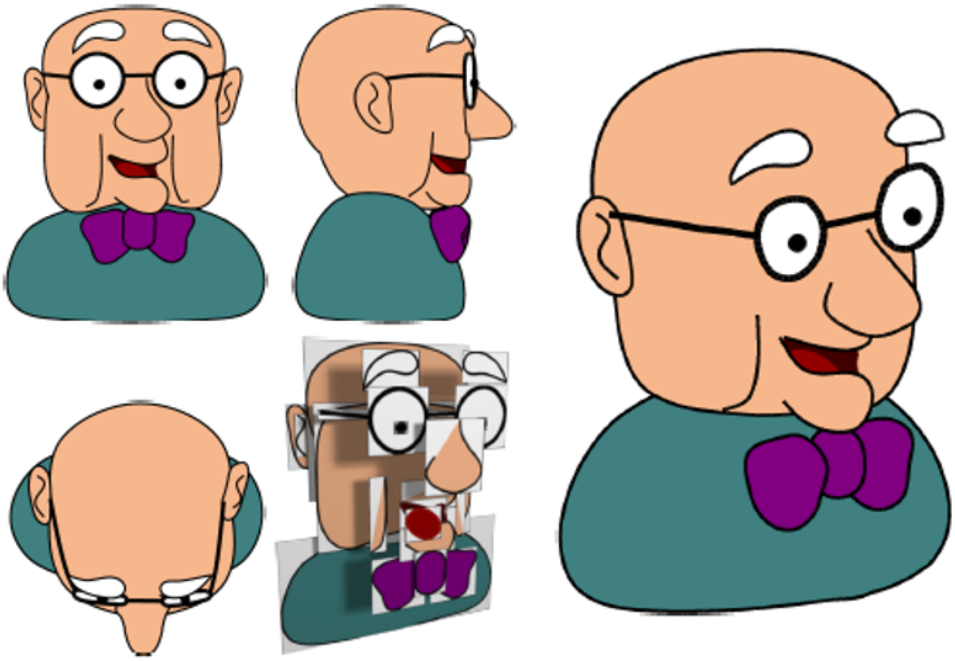


図 2.6　Rivers らの手法。様々な視点からの見た目をパーツごとにベクトルデータで作成し、視点に応じてこれらのパーツを変形させることで三次元空間での操作を可能にしている。画像は文献 [37] より引用。

図 2.7　Yeh らの手法。二次元モデルでオブジェクトの表面と裏面を作成し、これらの情報を用いることで疑似的に三次元モデルのような表現を可能にしている。画像は文献 [48] より引用。

2.3　カートゥーン調のキャラクタのためのアニメーション表現

疑似三次元モデルのキャラクタとして手描きのカートゥーンアニメーションが挙げられる。カートゥーンアニメーションではキャラクタの見た目が誇張して表現されるが、キャラクタの動きもまた誇張して表現される。CG におけるカートゥーンアニメーションのための様々な技法が提案されている。カートゥーンアニメーションのような動きを表現するために、オブジェクトの形状を変形させて動きを誇張する研究がある [13, 43, 29](図 2.8)。これらの手法では形状の見た目は誇張されるものの、動きのタイミングについては考慮していない。

カートゥーンアニメーションの中でもリミテッドアニメーションと呼ばれる表現手法では、動きを強調したり省略したりすることでメリハリのある動きを表現できる手法である。このように動きのタイミングを調整する研究は三次元キャラクタアニメーションの分野でも多く研究が行われている。 [21, 31, 26]。これらの手法はキーフレームあたりの時間を変化させることで動きのタイミングを調整し動きの速さを誇張しているが、リミテッドアニメーションのようにフレームを描かないことで動きの速さを強調するもの

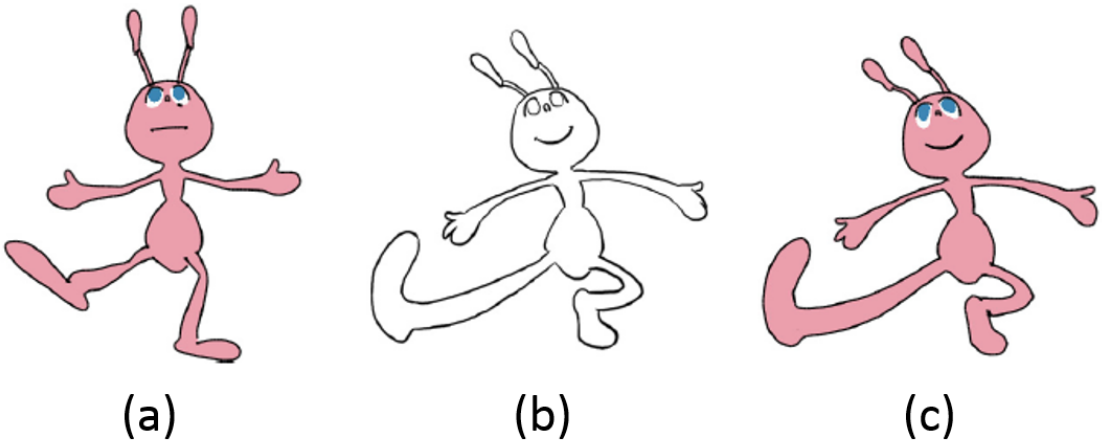


図 2.8　アニメーションの際の見た目の強調。(a) は元の誇張前の三次元モデル。(b) はデザイナによって描かれた誇張イメージイラスト。(c) は (a) を変形させて (b) に近づくように見た目を誇張したもの。画像は文献 [29] より引用。

ではない。一方、本論文の手法はモーションデータからフレームを削除するため、リミテッドアニメーションに近い効果が得られる。

リミテッドアニメーションの制作手法に着目した研究として、Morishima らはモーションキャプチャで得られたデータをリミテッドアニメーション風に変換する研究を行った。この手法では元のアニメーションより少ないキーポーズを選択するために、モーションカーブをキーポーズを頂点とする折れ線で近似し、折れ線の長さができるだけ大きくなるような最適化を行った [32]。この手法を用いることでモーションキャプチャのデータの動きを誇張することが可能となった。

第 3 章

疑似三次元モデルを用いたシーンの三次元モデリング

本章では疑似三次元モデルを用いたシーンの三次元モデリングについて解説する。疑似三次元モデルのシーンとして見下ろし型視点表示という表現手法に着目する。見下ろし型視点表示とは古くからコンピュータゲームで用いられている表現で、地形を表現するために使われている表現である。図 3.1 は見下ろし型視点表示を用いたゲームの例である。この表現は上から見たような表現であるが、同じ地形を三次元 CG で表現してもそのようには見えない表現である。見下ろし型視点表では真上からの見た目と正面からの見た目を同時に表示しているため、通常の三次元 CG とは見た目が異なるからである。しかし、我々はその表現からも地形の構造を知覚できる。本章では疑似三次元モデルのシーンである見下ろし型視点表示の地形を三次元モデルとして扱う方法について論じる。

3.1　課題と解決策

見下ろし型視点表示は真上からの見た目と正面からの見た目が混在するため、従来の三次元 CG では表現できない。従来の三次元 CG で同じ地形を表現するときには、真上と正面の情報を同じ比率で表現できないからである。本手法では見下ろし型視点表示を三次元モデルとして扱うために、見下ろし型視点表示から三次元構造を生成するモデリングルールを考案する。また、見下ろし型視点表示と三次元 CG の表現を滑らかにつなぐために、立体の形状を視点に応じて歪める手法も提案する。

図 3.1　見下ろし型視点表示を用いたゲーム作品。 真上からの見た目と正面からの見た目が混在しているが、地形や建物の形状が知覚できる。図は文献 [7]（左）、文献 [6]（右）から引用。

3.2　三次元モデリング手法

3.2.1　タイルベースインタフェースを用いた三次元モデリング

見下ろし型視点表示を三次元モデルとして扱うためには、見下ろし型視点表示から三次元モデリングを行う必要がある。本章では、そのためのモデリングルールの提案とモデリングインタフェースの実装を行う。図 3.2 は本章で開発したモデリングツールのインタフェースである。ユーザは見下ろし型視点表示での地形のモデリングを行うことができる。地形は正方形のタイルをカンバスに並べることで表現する。ユーザが回転操作を行うときに、システムは三次元モデリングを行い、見下ろし型視点から三次元 CG の表現へと切り替えることができる。この切り替えはシームレスに行うことができる。生成された三次元モデルはボクセルモデルとして表現される。本手法では緑、茶、青、赤、ベージュの 5 種類のタイルを入力として用い、それぞれのタイルにはエッジを入れることができる（図 3.3）。次節では見下ろし型視点表示からの三次元モデリングのルールについて述べる。

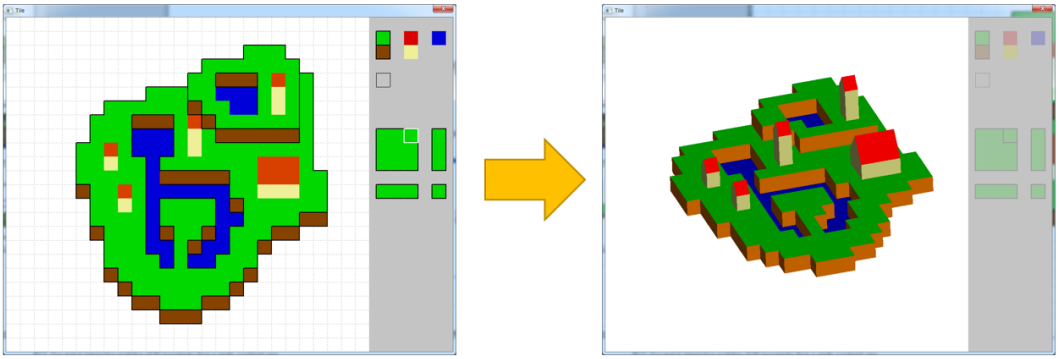


図 3.2　モデリングインタフェース。ユーザは正方形のタイルをカンバスに並べることで見下ろし型視点表示の地形を作成し（左）、回転操作を行うとシステムが三次元モデリングを行い、三次元モデルとしての操作が可能になる（右）。

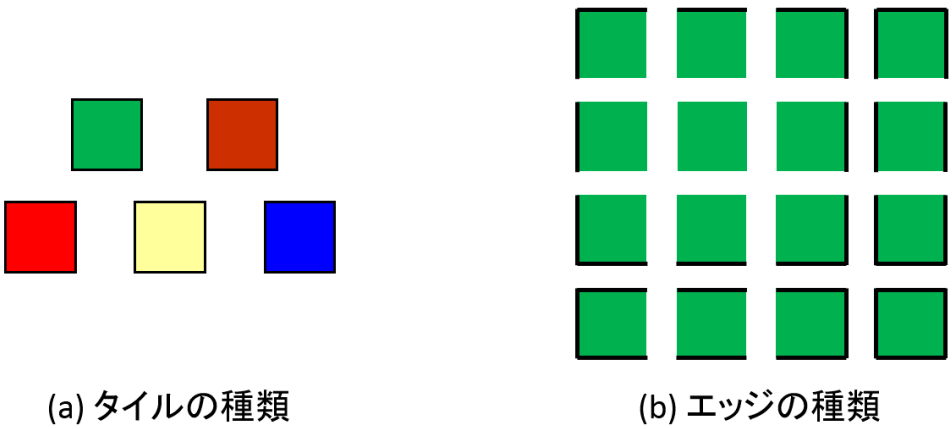


図 3.3　タイルの種類とエッジの種類。(a) は本手法で用いるタイルの色を表す。緑色と茶色のタイルを組み合わせることで地形を作成できる。赤色とベージュ色のタイルを組み合わせることで家を作成できる。緑色と茶色のタイルで作った地形に青色のタイルを組み合わせることで池を作成でき、地形を掘り下げることができる。(b) はタイルに付けることができるエッジのパターンを表す。エッジの有無で生成される形状が異なる。

3.2.2　基本地形の生成

緑色のタイルと茶色のタイルの組み合わせで基本地形を生成する。見下ろし型視点表示で立方体の表現は真上から見た正方形の形と正面から見た正方形の形を組み合わせて表現される。本手法ではこれを基準にルールを考案する。緑色のタイルで上面の形を作り、茶色のタイルで前面の形を作るように配置すると三次元地形が作成される。見下ろし型視点表示では垂直方向の情報が高さか奥行きを表しているが、本手法のシステムでは高さの情報として垂直方向に連結する茶色のタイルの数、奥行きの情報として垂直方向に連結する緑色のタイルの数を使うことで、垂直方向の情報を判定する。基本地形は箱上の形状が生成され、その箱の幅と高さと奥行きをそれぞれを W, H, D とし、見下ろし型視点表示において水平方向に連結する緑色のタイルの数を Nw_{green}、垂直方向に連結する茶色のタイルの数を Nh_{brown}、垂直方向に連結する緑色のタイルの数を Nh_{green} とすると、$W = Nw_{green}, H = Nh_{brown}, D = Nh_{green}$ となる（図 3.4）。基本立体作成の後、緑色のタイルで作られた閉領域がひとつのグループにまとめられる。高さはグループ内で一番高い高さを基準にするため、高さが低い基本立体は宙に浮いたような状態になる。また、図 3.5 の左側の物体は平たい箱の細長い箱が乗っているように見えるため、システムは初めに平たい箱と細長い箱を作り、平たい箱に細長い箱が乗るように細長い箱の高さを修正する。図 3.5 の右側にある物体は、何にも乗っていないので、左側の物体に比べ低い位置に生成されている。

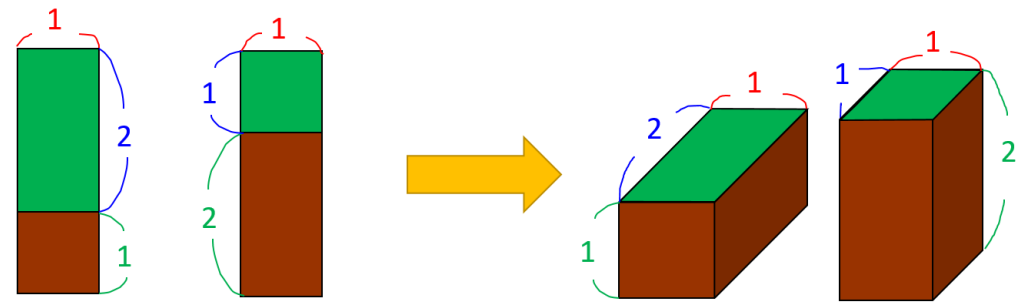


図 3.4　基本地形の生成。基本地形は箱上の形状が生成される。見下ろし型視点表示において水平方向に連結する緑色のタイルの数は三次元の箱形状の幅を表し、垂直方向に連結する茶色のタイルの数は箱の高さを表し、垂直方向に連結する緑色のタイルの数は箱の奥行きを表す。

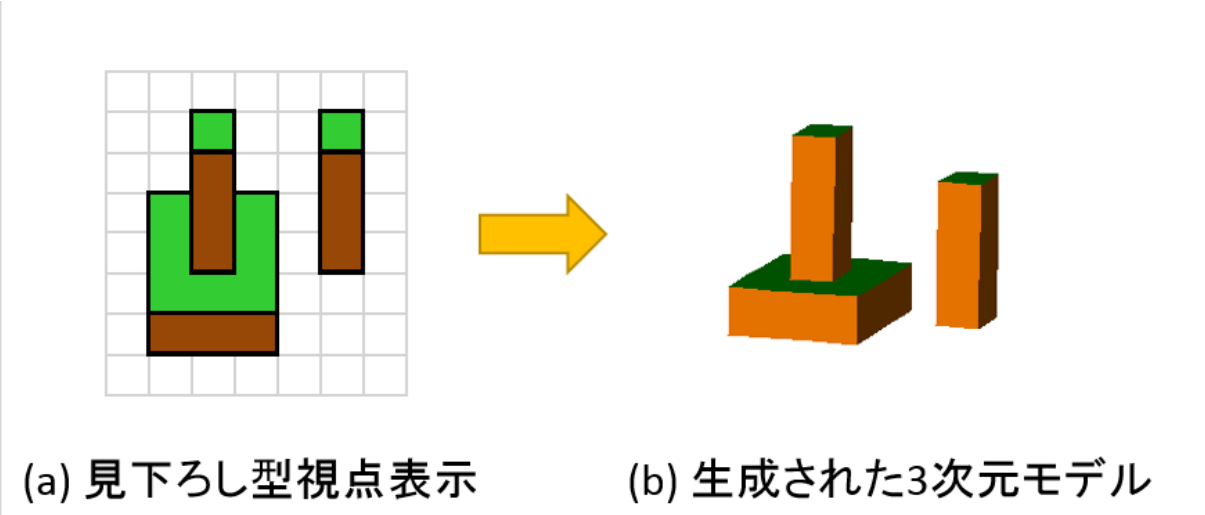


図 3.5　高さの修正。見下ろし型視点表示において左側の地形は、平たい箱のに細長い箱が乗っているように見えるため、細長い箱の高さに平たい箱の高さを加えることで、高さの修正を行う。右側の細長い箱は何にも乗っていないので、左側の細長い箱よりも低い位置にある。

3.2.3　エッジの有無による地形の変化

同じ色のタイル配置でもエッジの有無によって生成される形状が変化する場合について述べる。図 3.6 のタイル配置では階段状のオブジェクトを連想されるが、システムは緑色のタイルで作られた閉領域をひとつのグループとし、その高さを同一として扱うため、高さが低い基本立体は宙に浮いてしまい、連想される形状とは異なる形状が生成される。そこで緑色のタイルにエッジを入れることで、閉領域を構成する緑色のタイルの連結を切り、それぞれの基本立体が地に着くようになるため、階段状の地形を生成することが可能になる。

図 3.7 はエッジにより閉領域が分割され、2 つのグループが生成された場合の形状の変化である。エッジがない場合は、穴が空いた平たい箱形状が生成されるが、エッジのあるタイルを使い図 3.7 のように配置することで、2 つのグループに分けられ、平たい箱と小さい立方体が生成される。小さい方の立方体は平たい箱の上に乗っているように見えるため高さが修正され、生成される三次元形状も平たい箱の上に小さな立方体が乗っているような形状になる。また、図 3.7 を見下ろし型視点表示で見ると、小さな立方体の後ろの情報は遮蔽されているためわからない。本手法ではそのように遮蔽された部分が他の

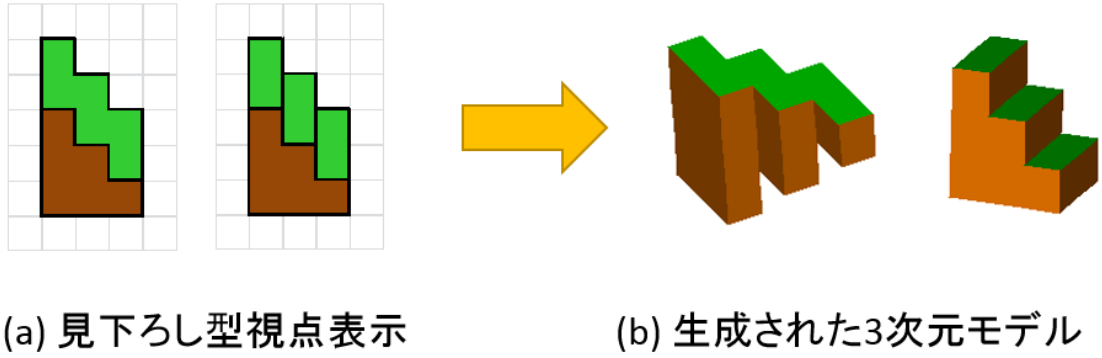


図 3.6　階段状の形状の作成。タイルの色の配置は同じでもエッジの有無により生成される形状が異なる。エッジにより緑色のタイルで作られる閉領域の形を変えることで、それぞれの基本立体が独立し地に着くようになり、階段状の地形の生成が可能になる。

グループの緑色のタイルと閉領域を作り得るなら、そのグループに連結するように形状を修正し、遮蔽部分の補完を行っている。

図 3.8 は茶色のタイルのエッジによる形状の変化である。茶色のタイルもまたエッジにより生成される立体の形状が異なる。茶色のタイルで構成される閉領域をひとつのグループとし、エッジによりグループが分割される。また、図 3.8(a) の右側の図形では分割された茶色のタイルは後方の茶色のタイルと連結しているように見えるため、生成される立体も後方の茶色の部分と連結する。

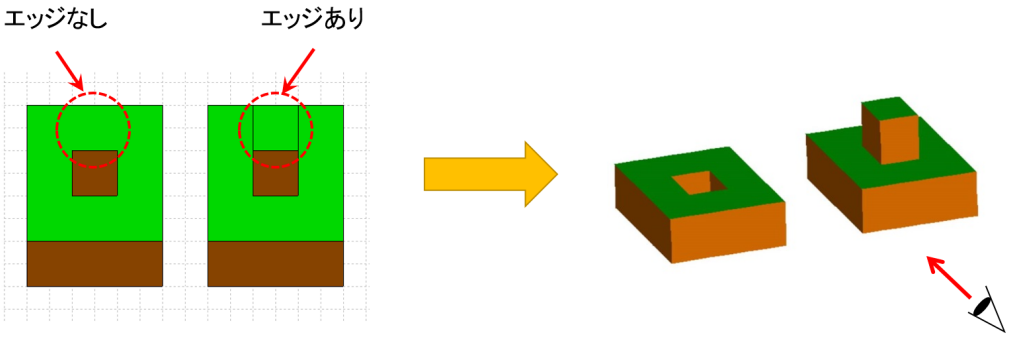


図 3.7　凹形状と凸形状。タイルの色の配置は同じでもエッジの有無により生成される形状が異なる。(a) の左側の地形ではエッジがないため一つのグループとして扱われ (b) の左側の形状のように穴が空いた箱のような形状が生成される。(a) の右側の地形では緑色のタイルで構成される閉領域のグループがエッジにより 2 つに分けられるため、(b) の右側の形状のように平たい箱の上に小さな立方体が乗っているような形状が生成される。

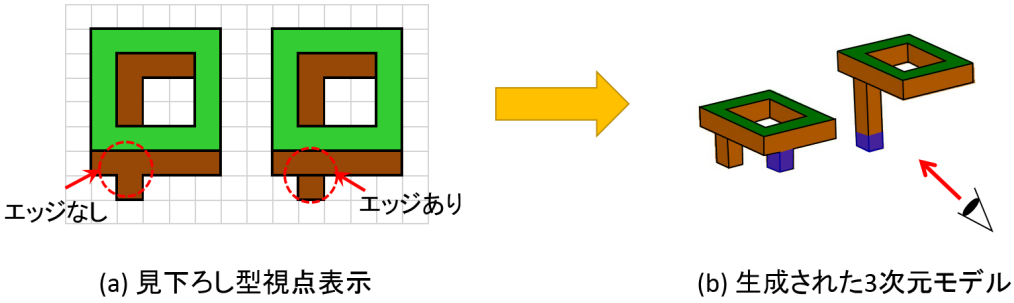


図 3.8　茶色タイルにおけるエッジの有無での変化。(a) の左側ではエッジがないため、茶色のタイルで構成される閉領域は一つのグループとして扱われる。(b) では茶色のタイルのエッジにより閉領域のグループが分割され、後方の茶色のタイルのグループと連結されるような形状が生成される。(b) の青色の部分がエッジにより形状が変化する部分である。

3.2.4　池の作成

青色のタイルを使うことで池のような地形を作成できる（図 3.9）。緑色のタイルと茶色のタイルでできた基本地形に池を追加するようにタイルを配置すると地形を掘り下げることができる。深さのある池を作るためには見下ろし型視点表示において垂直方向に上から、緑色、茶色、青色、緑色の順で連続してタイルを配置する必要がある。池の深さは垂直方向に連結する茶色のタイルの数で決められる。図 3.9 は池の作成例である。

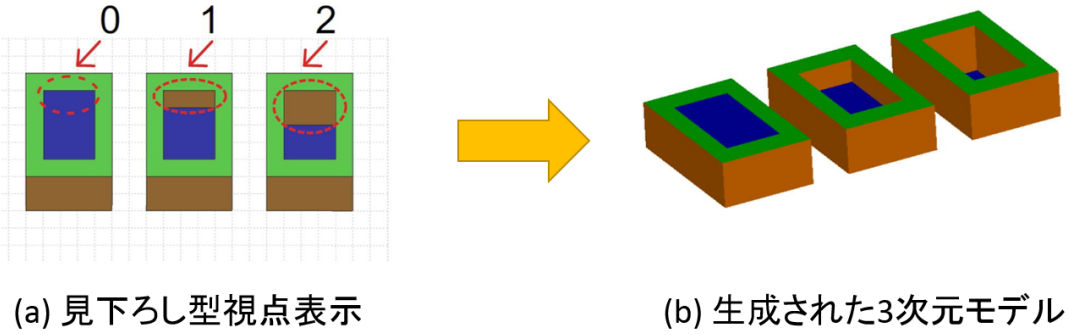


図 3.9　青色のタイルを用いて生成される池の地形。見下ろし型視点表示において垂直方向に連結する茶色のタイルの数が、生成される池の深さを表している。

3.2.5　家の作成

赤色のタイルとベージュ色のタイルを組み合わせることで家のような形状を作ることができる（図 3.10）。本手法のインタフェースでは、見下ろし型視点でのベージュ色のタイルが家の壁の部分を表し、赤色のタイルが家の屋根の部分を表している。本手法ではこのようなタイル配置から家の形状を作るために、家の形状を直方体と三角柱の組み合わせで表現する (図 3.11)。見下ろし型視点表示において家形状を作るときの垂直方向に連結する赤色のタイルの数を Nh_{red}、水平方向に連結する赤色のタイルの数を Nw_{red}、垂直方向に連結するベージュ色のタイルの数を Nh_{beige}、水平方向に連結するベージュ色のタイルの数を Nw_{beige} とすると、直方体の幅は Nw_{beige}、高さは Nh_{beige}、奥行きは Nh_{red} となる（図 3.12）。また、屋根形状を表す三角柱の底面の三角形の底辺の長さは Nh_{red}、三角柱の高さは Nw_{red} となり、三角柱の底面の三角形の高さは式 3.1 で求めら

れる（図 3.13）。

$$Height = \frac{Nh_{red}}{\sqrt{2}} \tag{3.1}$$

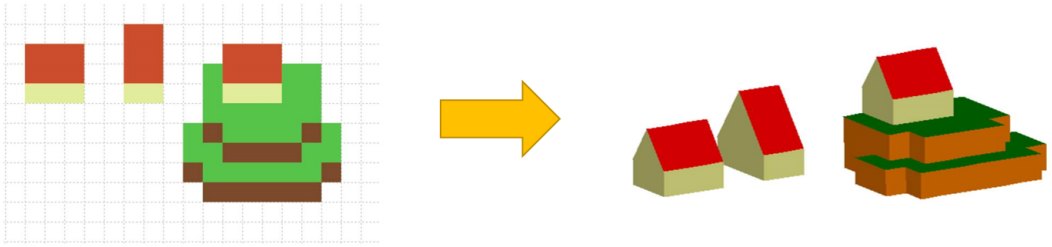


図 3.10　赤色とベージュ色のタイルを組み合わせて作る家形状。赤色のタイルは家の屋根の部分を表し、ベージュ色のタイルは家の壁の部分を表している。家形状は三角柱と直方体を組み合わせて形成される。また、ベージュ色のタイルの垂直方向下側に基本地形の緑色のタイルがあるときは、その緑色のタイルで生成される地形のグループの高さを適用することで、(b) の右側の図の地形のように基本地形の上に家が乗っているような形状が生成される。

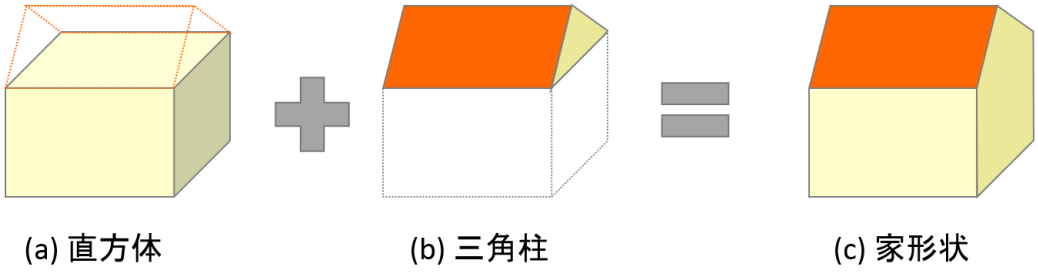


図 3.11　家形状の作成。家形状は直方体と三角柱を組み合わせることで生成される。

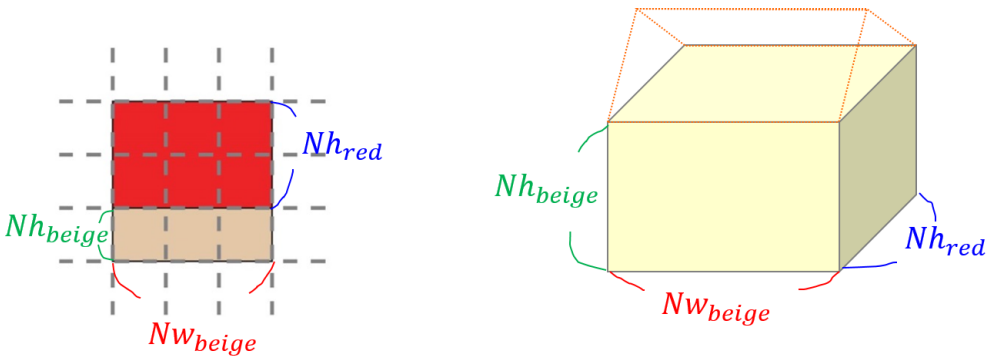


図 3.12　家の土台の作成。家の土台は直方体で表現される。直方体の幅は、見下ろし型視点での水平方向に連結するベージュ色のタイルの数 Nw_{beige}、直方体の高さは垂直方向に連結するベージュ色のタイルの数 Nh_{beige}、直方体の奥行きは垂直方向に連結する赤色のタイルの数 Nh_{red} より求められる。

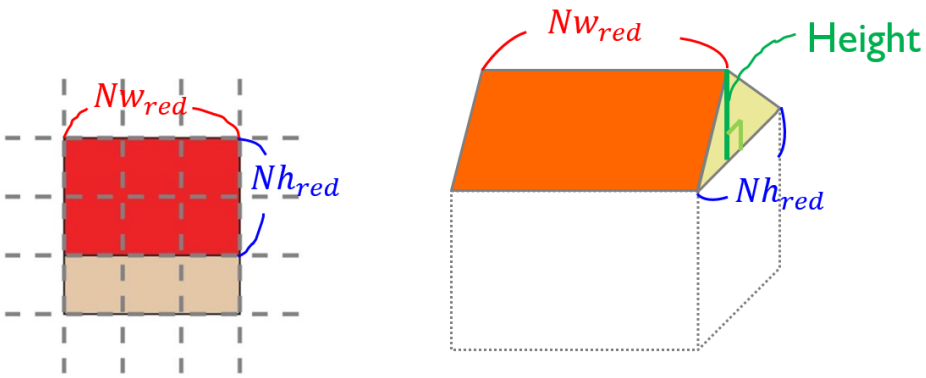


図 3.13　家の屋根の作成。家の屋根は三角柱で表現される。三角柱の底面の三角形の底辺の長さは見下ろし型視点表示で垂直方向に連結する赤色のタイルの数 Nh_{red} であり、底面の三角形の高さは式 3.1 で求められる。三角柱の高さは見下ろし型視点表示において水平方向に連結する赤色のタイルの数 Nw_{red} から求められる。

3.2.6　見下ろし型視点表示と三次元表示の滑らかな遷移

見下ろし型視点表示は真上からの見た目と正面からの見た目が混在しているため、通常の三次元 CG の表示方法では表現できない。立方体の形状を見下ろし型視点で表示すると、図 3.14(b) のようになり、通常の三次元 CG の表現手法で表現すると図 3.14(c) のようになる。見下ろし型視点表示と三次元 CG での表示を滑らかにつなぐために、本手

法では視点に応じて地形形状を歪めるようにした。図 3.14(a) のように視線ベクトルと地面の角度を θ とする。この角度に応じて変形率 R を求める。R は式 3.2 で求められる。変形は、地形形状を X 軸を中心に 45° 回転させた後、Y 軸方向に R 倍のスケーリングを行い、X 軸を中心に $-45°$ 回転させることで実現している。このようにすることで、θ が 45° 付近の時は見下ろし型視点表示に近い比率で表示し、0° に近いときは従来の三次元 CG での表示に近くなり、回転操作を行った時にスムーズに見下ろし型視点表示から三次元 CG での表示に切り替えることが可能となる。図 3.15 は見下ろし型視点表示と三次元 CG を滑らかにつなぐ例である。左上の図が見下ろし型視点表示でそれ以外の図は三次元 CG による表示である。視点に応じて形状を歪めているため、見下ろし型視点表示と三次元 CG の表示が滑らかにつながっていることがわかる。

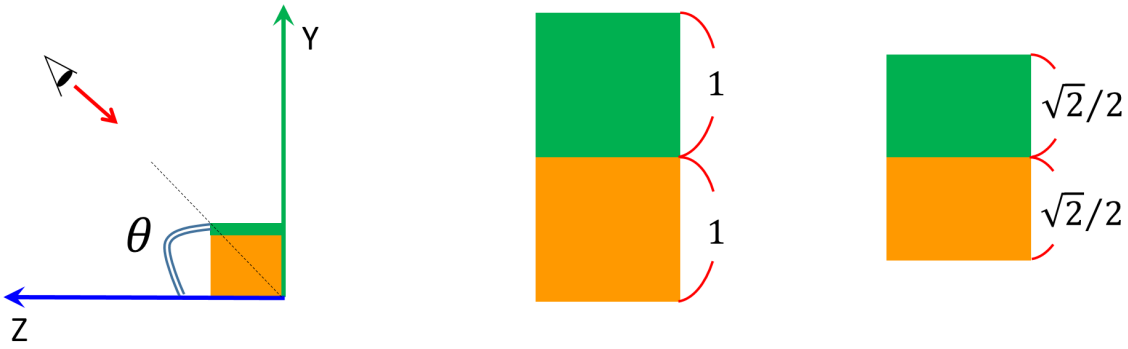


図 3.14　見下ろし型視点表示と従来の三次元 CG での表示の違い。斜め 45° の視点から立方体を見た場合、見下ろし型視点では真上からの見た目と正面からの見た目を同じ比率で表示 3 するため正方形が 2 つ連結しているように見える。平行投影を用いた三次元 CG では (c) のように水平方向の長さが (a) に比べ短くなっている。

$$R = \begin{cases} \frac{(\sqrt{2}-1)|\theta|}{45} + 1 \\ (for\ 0 \leq \theta < 45, -90 < \theta < -45) \\ \frac{(\sqrt{2}-1)(90-|\theta|)}{45} + 1 \\ (for\ 45 \leq \theta < 90, -45 \leq \theta < 0) \end{cases} \tag{3.2}$$

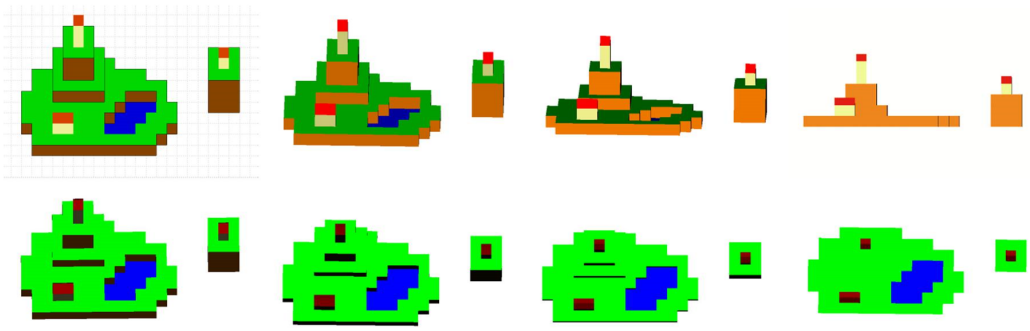


図 3.15　見下ろし型視点表示と三次元 CG での表示の滑らかな遷移。視点に応じて地形の形状を歪めることで、見下ろし型視点表示と三次元 CG での表示が滑らかにつながっている。左上の図が見下ろし型視点表示で、それ以外の図は三次元 CG での表示である。

3.3　結果と考察

本章では疑似三次元モデルのシーンの三次元モデリングについて述べた。疑似三次元モデルのシーンとして見下ろし型視点表示に着目し、見下ろし型視点を三次元モデルとして扱うためのモデリングツールを開発した。図 3.16 に本手法を用いた結果を示す。赤枠で囲まれている部分は見下ろし型視点表示で、青枠で囲まれている部分が生成された地形を三次元 CG で表現したものである。図 3.16 の上段は緑色と茶色のタイルで作られた複雑な地形が確認できる。図 3.16 の中段は赤色とベージュ色のタイルを使うことで生成された家と地形である。また、屋根のタイルの数を変えることで、生成される家の大きさが変わることも確認できた。図 3.16 の下段は青色のタイルを使って地形が掘り下げられ池が生成されている。図 3.17 は本手法で生成された地形にテクスチャマッピングを行ったものである。テクスチャの種類を変えることでより多くの表現が可能となる。このように見下ろし型視点表示という疑似三次元モデルを三次元モデルとして扱うためには、見下ろし型視点表示から三次元構造を作ることが必要である。また、視点に応じて地形形状を歪めることで見下ろし型視点表示と三次元 CG での表示を滑らかにつなぐことを可能にした。

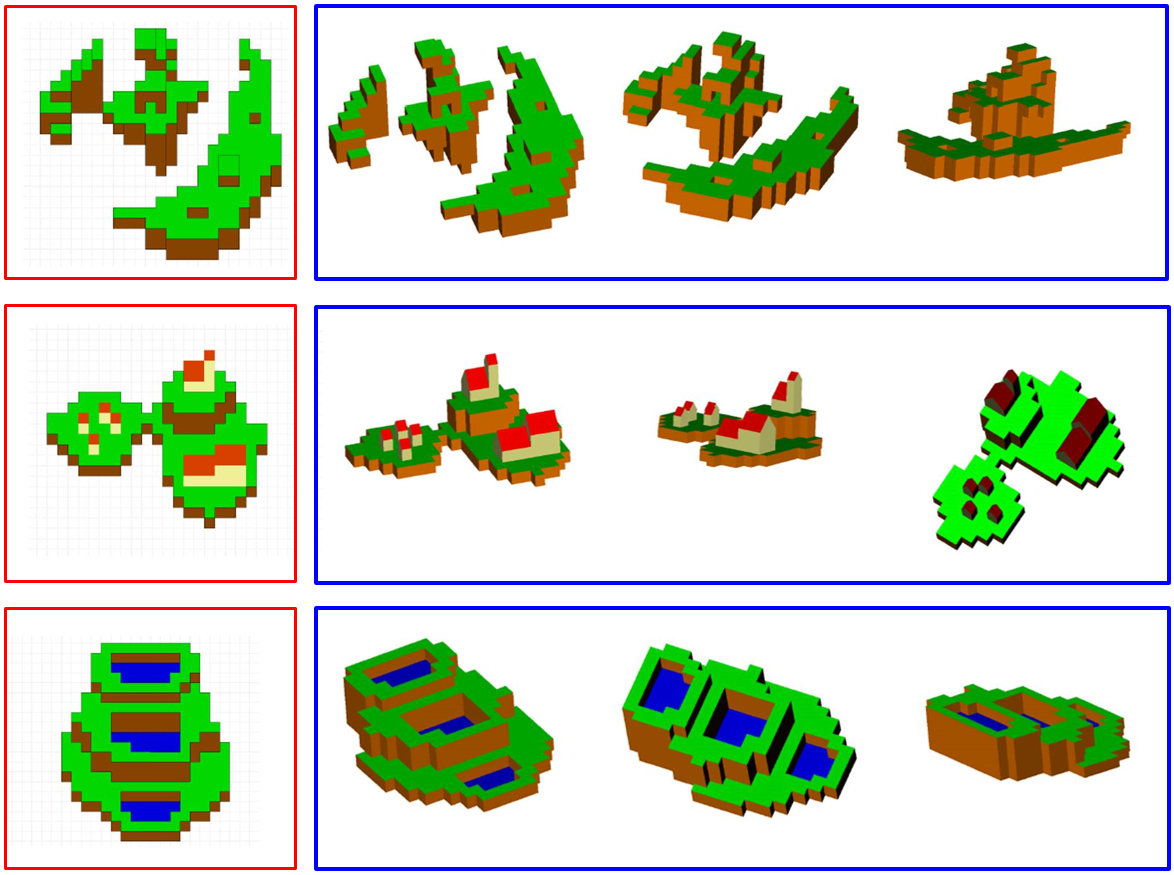


図 3.16　本手法により生成された結果。赤枠で囲まれている部分は見下ろし型視点表示での入力で、青枠で囲まれている部分は生成された三次元地形である。

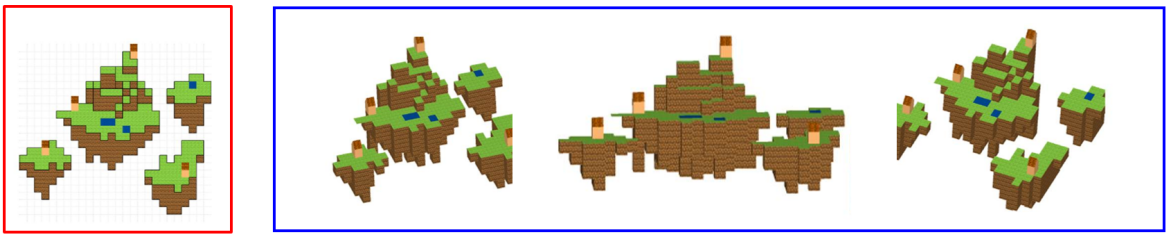


図 3.17　本手法により生成された結果にテクスチャマッピングを行ったもの。赤枠で囲まれている部分は見下ろし型視点表示での入力で、青枠で囲まれている部分は生成された三次元地形である。

第 4 章

疑似三次元モデルを用いたキャラクタの三次元モデリング

本章では疑似三次元モデルを用いたキャラクタを三次元モデルとして扱うための手法について述べ、それを実装したシステムについて解説する。また、本手法を用いて生成された結果についても述べる。

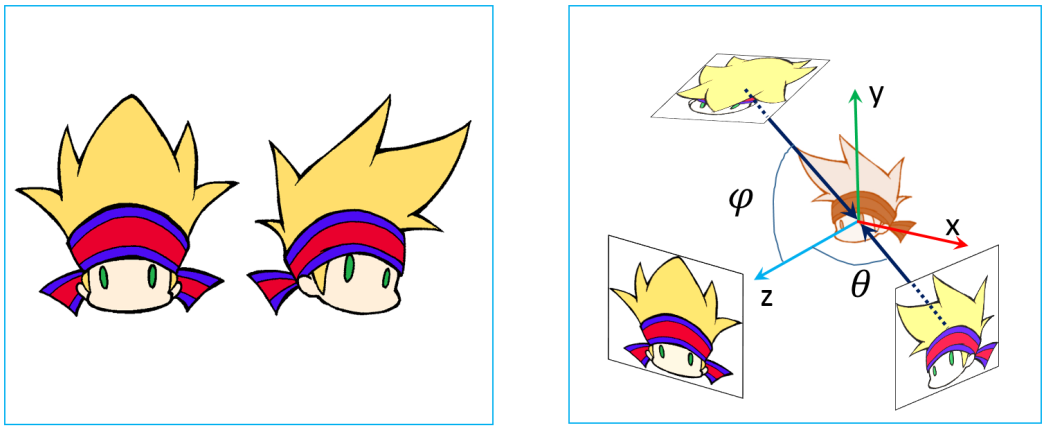
4.1　課題と解決策

疑似三次元モデルを用いたキャラクタは、特徴をわかりやすく表現するために形状が誇張して表現されることがある。例えば、髪型に特徴があるキャラクタの場合、正面から見た時の髪型と横から見た時の髪型が変わらないことがある。このキャラクタを三次元モデルで表現しようとすると正面からの見た目と側面からの見た目の形状に無理が生じてしまい、モデリング自体が非常に困難になってしまう。本章ではこのように従来の三次元モデリングを行う際に形状に無理が生じてしまうような形状であっても、これらを三次元モデルとして扱うことができる手法を提案する。具体的には、疑似三次元モデルのキャラクタを目や口、髪などのパーツに分けて扱い、それぞれのパーツをビルボードとして表現し、視点に応じてビルボードの形状を変形させることで、疑似三次元モデルを用いたキャラクタを三次元モデルとして扱うことができるようになる。

4.2　三次元モデリング手法

4.2.1　手法の概要

本章の手法では入力データとして正面図や側面図など、どの方向から描かれたか既知であるような 2 枚のイラストとその方向を用いる（図 4.1）。右手系の三次元直交座標系を次のように設定する。キャラクタの中心を原点とし、キャラクタは z 軸の正の方向を向いており、キャラクタから見て左および上方向をそれぞれ x, y 軸の正の方向とする。キャラクタを観察する視線方向と x 軸のなす角を θ、y 軸となす角を ϕ とし、キャラクタを正面から観察している状態を $(\theta, \phi) = (0, 0)$ とする。イラストは閉領域と輪郭線で構成され、閉領域内の色は単色で輪郭線の色は黒色とする。



(a) 異なる方向から描かれた2枚のイラスト　　(b) イラストが描かれている方向

図 4.1　入力データ。本手法では異なる視点から描かれた 2 枚のイラスト (b) とそれが描かれている向き (b) を入力とする。入力画像のイラストは閉領域と輪郭線で構成され、閉領域内の色は単色で輪郭線の色は黒色とする。イラストが描かれている向きに関しては、キャラクタを観察する視線方向と x 軸のなす角を θ、y 軸となす角を ϕ とし、キャラクタを正面から観察している状態を $(\theta, \phi) = (0, 0)$ とする。

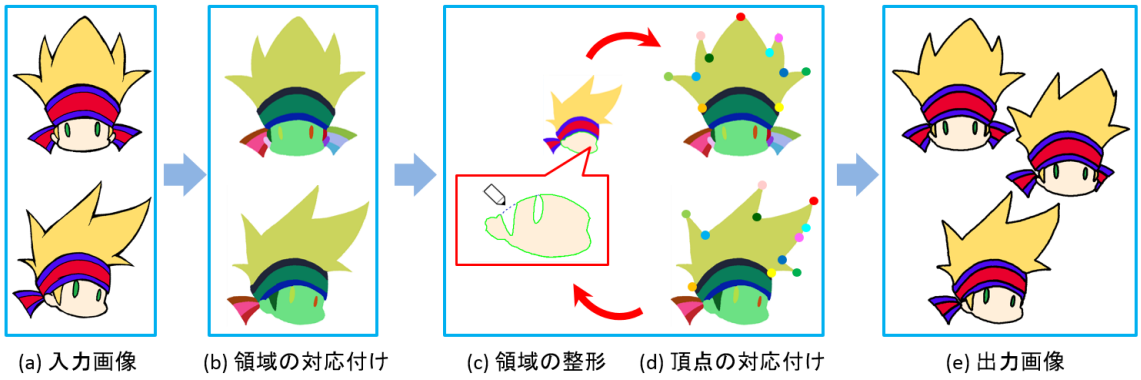


図 4.2　手法の概要。

図 4.2 に本手法の概要を示す。本手法ではイラストを閉領域に分割し (図 4.2(a))、2 つのイラストの間で閉領域を対応付け (図 4.2(b))、対応付けられた閉領域をビルボードとして三次元空間に配置する。その三次元位置は対応付けられた閉領域の重心位置と各イラストの視線方向から求められる。ビルボードの表示は平行投影による。ビルボードの形状は、2 枚のイラストから得た閉領域の形状を線形補間することで得られる。この補間のために、ユーザは閉領域の形の修正や特徴点の対応付けを行う (図 4.2(c),(d))。これらの処理を行い三次元モデルのビルボードを作ることで、奥行きを考慮した見た目の変化を表現できる。

4.2.2　領域分割と領域の形状整形

最初に入力画像に対して flood fill アルゴリズムを用いて閉領域に分割する (図 4.2(a))。このとき、輪郭線は一定の太さを持っているため、輪郭線を含まずに閉領域をビルボードとして配置すると閉領域間に隙間ができてしまう (図 4.3(a))。そこで輪郭線の領域を分割された閉領域のいずれかに統合する処理を行う。これは Sýkora らの手法 [39] と同様の手法を用いる。具体的には輪郭線のピクセルデータと各閉領域との距離を求め、距離が一番近い閉領域に輪郭線ピクセルを統合する。図 4.3(b) は輪郭線部分のピクセルが閉領域に統合されたものである。

また、小さい閉領域が大きい閉領域に内包される場合は大きい閉領域に穴が空くが、穴が空いたまま補間すると穴まで補間されてしまう。これを防ぐために、穴が空いている部分を埋める処理を自動で行う。図 4.2(c) は目の閉領域によって顔の閉領域に空いた穴を埋めている例である。また、イラスト上では 2 つの閉領域が互いに隣接していても、スクリーン投影時にビルボードの奥行きの違いによって閉領域間に隙間が生じることがあ

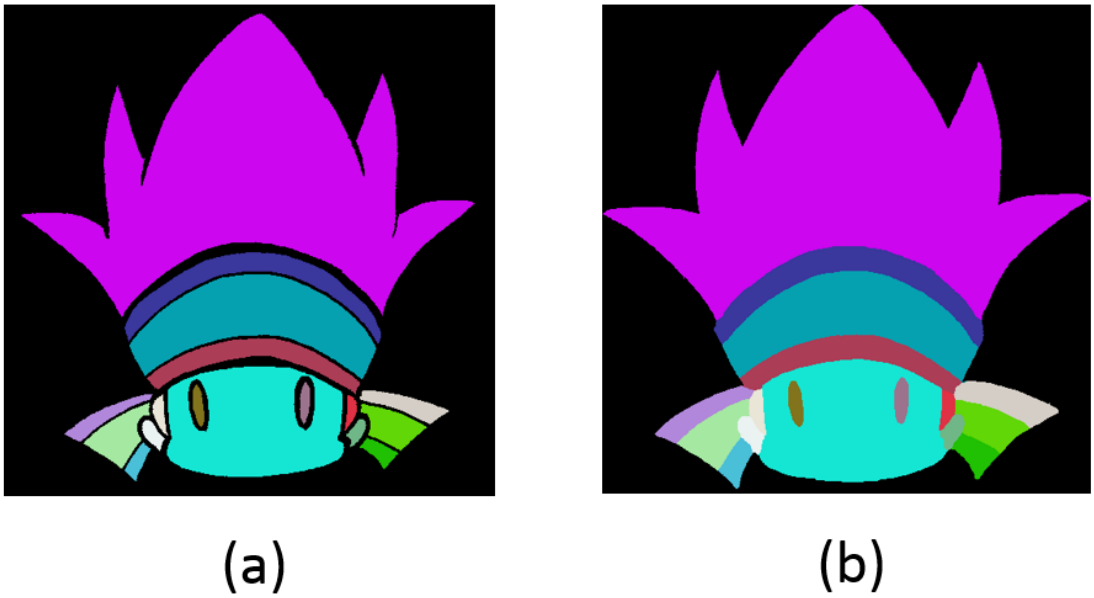


図 4.3　輪郭線部分の統合。(a) は輪郭線以外の部分の閉領域。輪郭線があったところに隙間が生じている。(b) は輪郭線部分を他の領域に統合したもの。輪郭線領域のそれぞれのピクセルと各閉領域の距離を計算し、距離が最も近い閉領域に統合する。

る。このような隙間をなくすために、ユーザが遮蔽された閉領域の形状のストロークを描くことでその形状を変形することも可能である。

4.2.3　類似度による領域の対応付け

領域分割の後、一方のイラストの各閉領域に対し、他方のイラストの中で類似度が最も高い閉領域を対応付ける (図 4.2(b))。閉領域の類似度の計算方法として Liu らの手法がある [30]。彼らの手法では閉領域の色、形状、重なっている部分の面積を考慮して類似度が計算されるが、片方の閉領域が他方の閉領域に完全に内包される場合に類似度が最大となる。このことが本手法では問題となり得る。例えば、視点の変更によって目の位置がずれて顔の閉領域に完全に内包され、顔と目の類似度が最大となり、誤って対応付けられてしまう。そこで本手法では Liu らの手法 [30] を改良し、閉領域 a, b に対する類似度 $s_{a,b}$ を次式 (4.1) によって計算する。

$$s_{a,b} = \mathcal{J}_{a,b}\, e^{-w_x|x_a - x_b|}\, e^{-w_y|y_a - y_b|}\, e^{-\frac{|S_a - S_b|}{(S_a + S_b)/2}} \tag{4.1}$$

この式のうち $\mathcal{J}_{a,b}$ のみが Liu らのものと共通している。これは色に関する類似度であり、閉領域の色の差が一定の範囲内にあるかを判定している。

$$\mathcal{J}_{a,b} = H[T_c - C_{a,b}] \tag{4.2}$$

ここで $C_{a,b}$ は RGB 色空間における閉領域 a, b のベクトル $\mathbf{q}_a, \mathbf{q}_b$ のユークリッドノルムであり $C_{a,b} = ||\mathbf{q}_a - \mathbf{q}_b||$ である。T_c はユーザ定義の定数であり、本研究では 0.3 を用いた。$H[n]$ は Heaviside ステップ関数であり、n が 0 以上のとき 1 でそれ以外は 0 である。$e^{-w_x|x_a-x_b|}\, e^{-w_y|y_a-y_b|}$ は位置の類似度を表している。この類似度はキャラクタを観察する視線方向と位置の変化を考慮して求められる。閉領域 a, b のバウンディングボックスの中心をそれぞれ (x_a, y_a), (x_b, y_b) とする。視線方向と x 軸のなす角 θ が変化しない場合、つまり視線が上下にのみ動く場合はスクリーン座標系での x 成分はほとんど変化しないため、特に $|x_a - x_b|$ がより小さい場合に類似度を高くする。一方、視線方向と y 軸のなす角 ϕ が変化しない場合は、特に $|y_a - y_b|$ がより小さい場合に類似度を高くする。これを実現するために、重み w_x, w_y を角度に応じて調整する。w_x は $\theta = 0$ のとき 1.0、それ以外のとき 0.5 とし、w_y も ϕ に関して同様に定める。$e^{-\frac{|S_a-S_b|}{(S_a+S_b)/2}}$ は閉領域の大きさについての類似度である。領域 a, b のそれぞれの面積を S_a, S_b とするとき、S_a, S_b が互いに近ければ類似度が大きくなる。正規化のために面積の平均で除算している。

上記の類似度に基づき、一方のイラストの各閉領域に対し、他方のイラストの中で最も類似度の高い閉領域のペアを対応付ける。この手法を用いて対応付けを行った結果を図 4.4 と表 4.1 に示す。図 4.4 において (a) は入力画像、(b) は Liu らの手法による対応付け、(c) は本手法による対応付けの結果である。閉領域の対応付けが正しく行われている部分はグレー、間違っているところは赤、対応が取れなかった部分は青で示している。表 4.1 は、対応が存在する閉領域のうち、正しく対応付けられた正解数を示している。少年の例では、Liu らの手法は、顔と耳の閉領域の類似度が誤って高くなってしまい対応付けに失敗している。本手法においては両耳と右側の鉢巻と髪の毛の対応が存在しないが、これらに対応する閉領域は存在しないため、対応が取れないことが正解である。イヌの例では、Liu らの手法は顔と口の閉領域の類似度が誤って高くなってしまい正しく対応付けられていない。目に関しては両手法とも間違っており、左右が入れ替わって対応付けられている。ネコの例では、Liu らの手法は顔と口の閉領域が誤って対応付けられている。本手法ではネコの左の前足の対応が存在しないが、これは対応する閉領域がないので正解である。タヌキの画像においては、Liu らの手法ではボタンの対応が間違っているが本手法では正しく対応付けられている。対応付けが間違っている場合は、ユーザがインタラクティブに修正できる。また、遮蔽されていて対応が取れない閉領域に関しては、ユーザが遮蔽されている位置に閉領域の形状のストロークを描き、新しい閉領域を作成

することで対応付けを行う。

表 4.1　遮蔽されておらず対応付け可能な閉領域に対する正解数。括弧内の数字は正解率。

画像	対応付け可能な閉領域数	Liu らの手法	本手法
少年	11	9 (0.82)	11 (1.00)
イヌ	9	2 (0.22)	5 (0.56)
ネコ	12	8 (0.67)	11 (0.92)
タヌキ	23	15 (0.65)	17 (0.74)

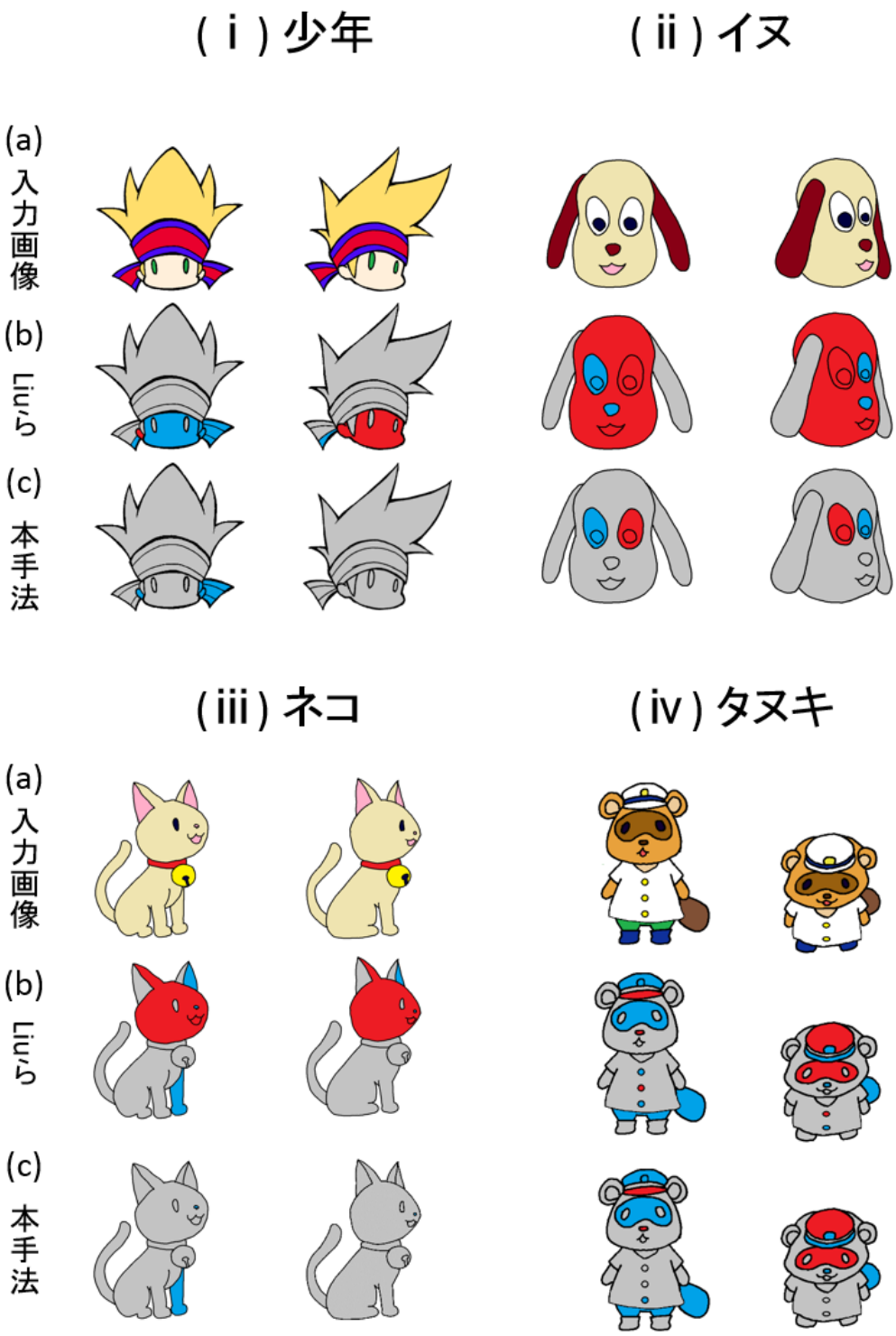


図 4.4　類似度による対応付けの比較。閉領域の対応付けが正しく行われている部分はグレー、間違っているところは赤、対応が取れなかった部分は青で示している。

4.2.4　ビルボードの三次元空間での位置の推定

異なる視点から見た閉領域の対応が取れれば、閉領域をビルボードとして表現したときの三次元空間での位置 $\mathbf{p}$ を推定できる (図 4.5)。この手法は Rivers らの手法 [37] を用いる。この手法の詳細について著者に確認したところ、次の Algorithm 1 を用いているとの回答があった。

この手法を用いることで各閉領域のビルボードの三次元空間での位置 $\mathbf{p}$ を求めることができるが、この座標をそのまま用いるとスクリーン座標系におけるビルボードの位置と入力画像の閉領域の位置にずれが生じる。このずれをなくすために、対応付けられた 2 つの閉領域 a, b の重心から視線方向に伸びる半直線に $\mathbf{p}$ を射影することで各閉領域の最終的な三次元空間での位置 $\mathbf{p}_a$ と $\mathbf{p}_b$ を決める。イラスト間の補間を行うときは、$\mathbf{p}_a$ と $\mathbf{p}_b$ を線形補間して求める。本システムでは、ビルボードの奥行きを修正するためにユーザがインタラクティブに任意の奥行きを設定できる。

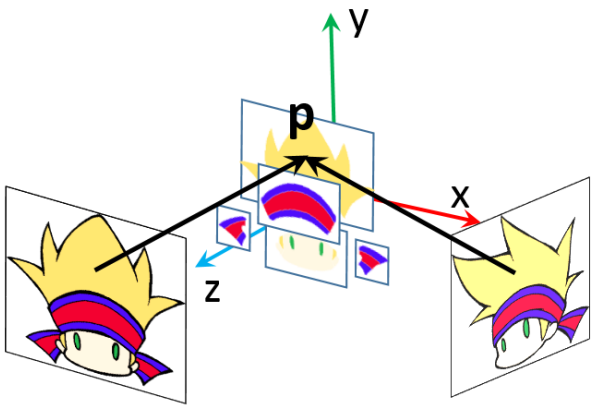


図 4.5　三次元空間でのビルボードの位置。

Algorithm 1 Calculate a 3D billboard position.

$\mathbf{p}_{sum} \leftarrow (0,0,0), \mathbf{p}_{current} \leftarrow (0,0,0)$
$N \leftarrow$ many times (e.g. 10,000)
for $i = 0$ to N **do**
  $N_V \leftarrow$ the number of views
  for $j = 0$ to N_V **do**
    $\mathbf{c}_j \leftarrow$ the center of bounding box of region R_j
    $\mathbf{l}_j \leftarrow$ the 3D line that passes through $\mathbf{c}_j$ and goes in the view direction of the view
    $\mathbf{p}_j \leftarrow$ the 3D position that projected $\mathbf{p}_{current}$ onto $\mathbf{l}_j$
    $\mathbf{p}_{sum} \leftarrow \mathbf{p}_{sum} + \mathbf{p}_{current}$
  end for
  $\mathbf{p}_{current} \leftarrow \mathbf{p}_{sum}/N_V$
end for
$\mathbf{p} \leftarrow \mathbf{p}_{current}$

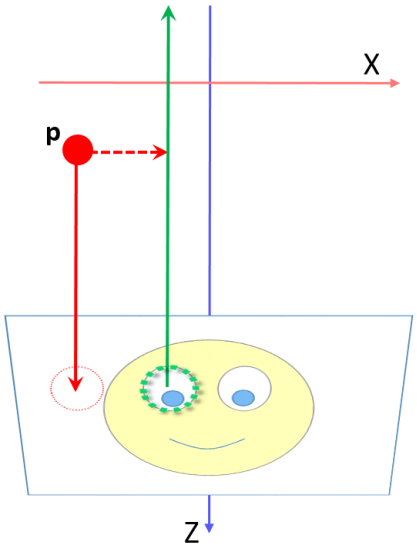


図 4.6　三次元空間での位置の修正。この図において目の領域の位置 $\mathbf{p}$ をそのままスクリーン座標に投影すると、元のイラストの目の位置とずれが生じる。これを防ぐためにイラストの目の領域の中心から視線方向に伸ばしたベクトル（図中の緑の矢印）に $\mathbf{p}$ を射影し、この位置を三次元空間での位置とすることで、元のイラストとのずれをなくすことができる。

4.2.5　領域の形状の特徴点の対応付け

対応付けと 3D 位置の決定により、各閉領域は三次元空間にビルボードとして配置される。ビルボードの形状は、2 枚の入力画像の各閉領域の輪郭の形状を線形補間したものである。本研究では閉領域の輪郭線上に特徴点を設定し、対応付けられた閉領域間で特徴点を対応付ける。これらの特徴点ごとに輪郭線を線分で区切り、その線分ごとに補間を行う。特徴点はユーザがインタラクティブに付与することができる。図 4.7 に特徴点の例を示す。なお、Baxter らの手法 [12] のように特徴点を自動で対応付ける手法があるが、遮蔽を伴う動きのように画像間での変化が大きい場合は自動で適切な対応付けを行うことは困難であり、対応付けの修正に手間がかかるため、本手法では特徴点の付与と対応付けはユーザ入力とした。

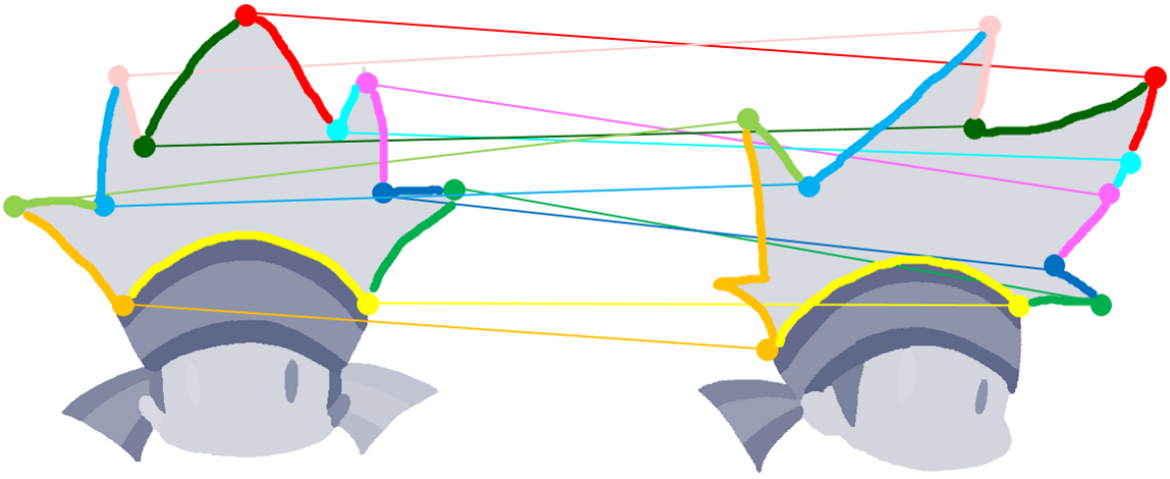


図 4.7　特徴点の対応付け。閉領域を特徴点ごとに線分に分割し、線分ごとに補間を行う。

4.3　結果と考察

本章では疑似三次元モデルを用いたキャラクタを三次元モデルとして扱う方法について述べた。具体的にはキャラクタを目や口などのパーツに分割し、パーツをビルボードとして三次元空間に配置し、視点に応じてビルボードの形状を変形させる手法である。また、それを実現するアプリケーションの開発も行った。アプリケーションは C++ と Qt Library を用いて実装した。実行環境として Intel Core i7-2760QM 2.40GHz CPU

の PC を用いた。図 4.8 は本手法を用いて生成したアニメーションの一部である。赤枠で囲んだイラストが入力となる画像データで、青枠で囲まれているイラストは本手法により生成されたフレームである。入力したイラストは、それぞれ正面画の他に、少年、イヌ、ネコのイラストでは $\phi = 0$ のまま θ のみ 45° 変化させたもの、タヌキのイラストでは $\theta = 0$ のまま ϕ のみ 45° 変化させたものを用いた。制作にかかった時間はいずれも 10 分から 50 分程度であり、その中でも領域の整形と奥行きの修正に大部分の時間を必要とした。少年のイラストでは髪や鉢巻の結び目が遮蔽されていく様子が表現できている。イヌのイラストでは左耳が顔に遮蔽される様子が表現できている。ネコのイラストでは左側の前足が遮蔽されていく様子が表現できている。タヌキのイラストでは黄色のボタンとズボンが遮蔽されていく様子が表現できている。

本手法においてキャラクタの一部が遮蔽されていく様子は表現できるが、キャラクタの輪郭線の表示に問題が生じる場合がある。本手法では閉領域の境界に黒い輪郭線を描いているが、この手法では元の輪郭線となるピクセルの座標しか保持しないため、輪郭線の太さや太さの変化の度合いを表現することができない。図 4.8 の入力画像では金髪の少年の髪の毛の閉領域の輪郭線の内側にも線が描かれているが、現在の実装ではこのような線を表現できない。今後はこのような線の情報の表現も必要であると考えられる。

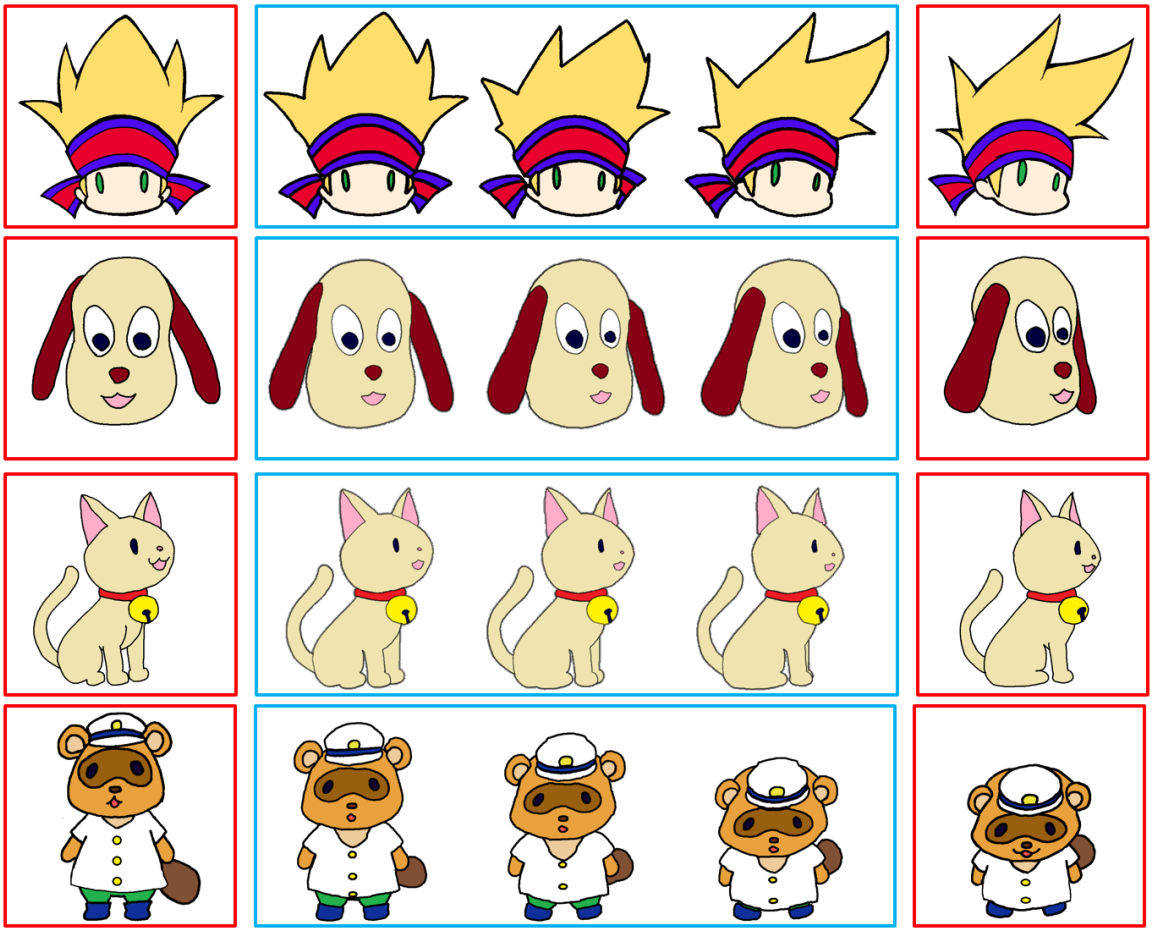


図 4.8　本手法により生成された画像の一部。赤枠で囲んだイラストが入力となる画像データで、青枠で囲まれているイラストは本手法により生成されたフレームである。

第 5 章

疑似三次元モデルのキャラクタのためのアニメーションの生成

5.1　はじめに

本章では疑似三次元モデルで表現されたキャラクタへのアニメーションの適用について述べる。前章では疑似三次元モデルのキャラクタとしてカートゥーン調のキャラクタに着目した。本章ではそのカートゥーン調のキャラクタのためのアニメーションについて論じる。従来の三次元モデルにおいてキャラクタのアニメーションを生成する際は、実際の人間の動きをモーションキャプチャでキャプチャしたものが用いられることが多い。一方、カートゥーンアニメーションではリミテッドアニメーションというスタイルのアニメーション表現が用いられることが多い。そこで、本手法では疑似三次元モデルのキャラクタのためのアニメーション生成としてリミテッドアニメーションの適用を試みる。リミテッドアニメーションとは、同じ絵を 2 コマあるいは 3 コマ連続して用いる「コマ撮り」(図 5.1) によって、毎秒 24 コマのアニメーションを構成する表現手法である。元々はコスト削減のために開発されたが、意識的に絵や動きに制限を加えた表現としても発展し特有のメリハリのある表現を可能にした。リミテッドアニメーションは手描きアニメーションのための手法であるが、三次元 CG の分野でもリミテッドアニメーションの表現手法が注目されてきている。

手描きアニメーションの手法である形状変形による誇張表現やコマ撮りの手法は、カートゥーン調のキャラクタの動きを表現するのに適している。一方で、カートゥーン調のキャラクタにモーションキャプチャのデータをそのまま適用することは適していない。このことに関して、著名なアニメーション映画監督の John Lasseter は次のように述べている [28]。

> *“Motion capture from human actors will always look realistic... for a human. But apply that motion to a chicken and it will look like a human in a chicken suit.”*

しかし、カートゥーン調の三次元キャラクタを扱った最近のゲームでは、制作コストの問題でモーションキャプチャのデータを用いることが多い。そこで本研究ではモーションキャプチャのデータを入力として、リミテッドアニメーションで用いられている表現手法の効果を模倣する手法を提案する。

リミテッドアニメーションの特徴として、本章では次の 2 つに着目する。一つはコマ撮りによって動きが少ないフレームを描かないため、細かい動きがないことである。もう一つは連続している動きの中間の絵を描かない「中無し」 [50] という手法を用いることで動きの速さの強調を行っていることである。本研究ではこれらの効果を模倣するために、モーションキャプチャのデータの細かい動きを低減し、動きの速さの強調を行うことを目的とし、動きが大きい部分のフレームを抜いたり、動きが小さい部分のフレームをコマ撮りにすることで、これらの効果を実現する。

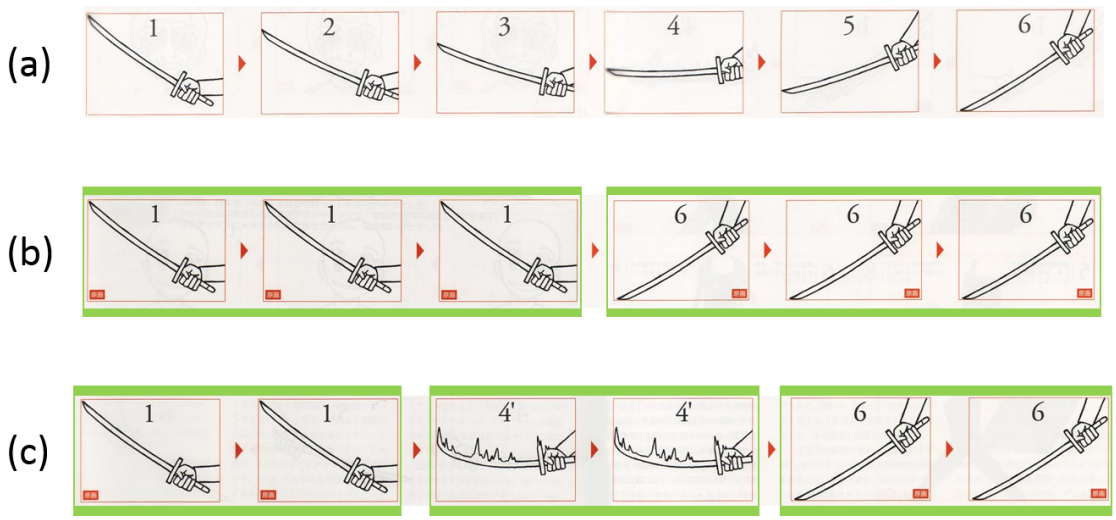


図 5.1　コマ撮り。(a) は全てのフレームの絵を描くフルアニメーションと呼ばれる手法。(b) と (c) はリミテッドアニメーションの手法でコマ撮りと呼ばれる。(b) は 3 コマ撮りと呼ばれ、同じ絵を 3 枚ずつ使ってアニメーションを構成する。同様に (c) は 2 コマ撮りと呼ばれ、同じ絵を 2 枚ずつ使う。図は文献 [50] から引用。

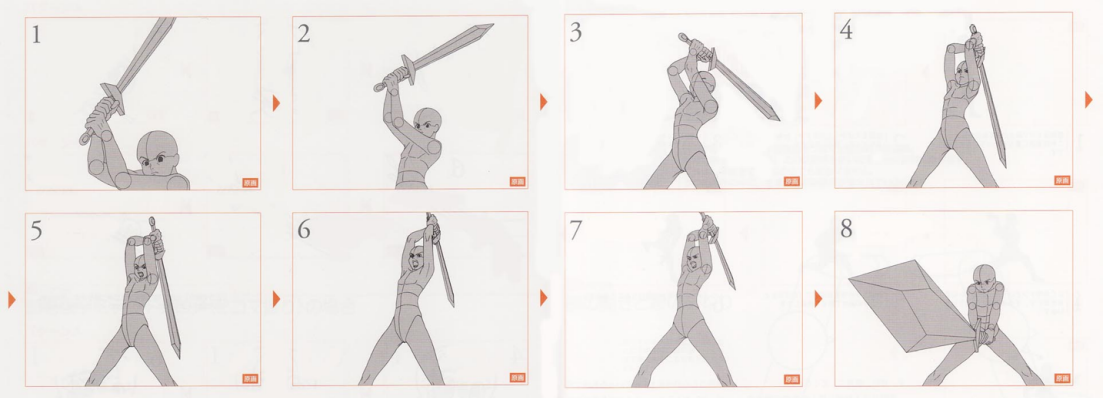


図 5.2　中無し。7 フレーム目と 8 フレーム目の間の動きを描かないことで動きの速さを強調している。図は文献 [50] から引用。

5.2　アルゴリズム

提案手法は、入力されたモーションキャプチャのデータに対し、動きの少ないコマを抜いて細かい動きを低減し、著しく速いコマを抜いて速さを強調する。本論文の手法では入力となるデータは「投げる」、「振る」といった一つの動作で構成されていると仮定し、それらを組み合わせた複合的な動作の場合は手動で分割している。

5.2.1　速さの定義

本手法では、入力となるモーションデータはフレーム数が N 枚でジョイントの数は n 個であるとする。フレームレートは毎秒 24 フレーム（24FPS）とし、フレーム間の時間間隔はすべて等しいものとする。さらに、k フレーム目（$k = 1, 2, \ldots, N$）でのジョイント i（$i = 1, 2, \ldots, n$）における座標を $\mathbf{p}_i^k \in \mathbb{R}^3$ とする。k フレーム目と $(k-1)$ フレーム目の各々のジョイント位置の二乗距離の総和 (SSD: the Sum of Squared Distances) を計算し、これを k フレーム目での速さと定義する。

$$SSD(k) = \sum_{i=1}^{n} \| \mathbf{p}_i^k - \mathbf{p}_i^{k-1} \|^2 \tag{5.1}$$

5.2.2　中無しによる速さの強調

中無しの効果を再現するために、入力データの動きが速い部分の速さを強調する処理を行う (図 5.3)。まず、SSD が最大となるような k_{max} 番目のフレームを求め、このフレームを削除する。フレームを削除すると、モーション全体のフレーム長が変わってしまうため、フレーム長を一定に保つために、変化が少ない部分にフレームを追加する。k_{min} 番目の SSD が最小となる場合、$(k_{min}-1)$ 番目のフレームをコピーして挿入する。同じフレームが 3 フレームより多くコピーされると動きが静止して見えるため、すでに 3 フレームコピーされたフレーム間にはコピーせず、SSD が次に小さいフレーム間にフレームを追加する。本手法では著しくモーションのタイミングが変わることを防ぐために、フレーム長を一定に保つようにしているが、フレーム長が変わってよい場合は、この処理は不要である。フレームの削除と挿入の後に、再び SSD を再計算し、これらの処理を繰り返し行うことで中無しのように速さを強調する効果が得られる。この処理の疑似コードを Algorithm 2 に示す。#DUPLICATED(k) は k フレーム目を複製して新たに挿入したフレーム数を意味する。

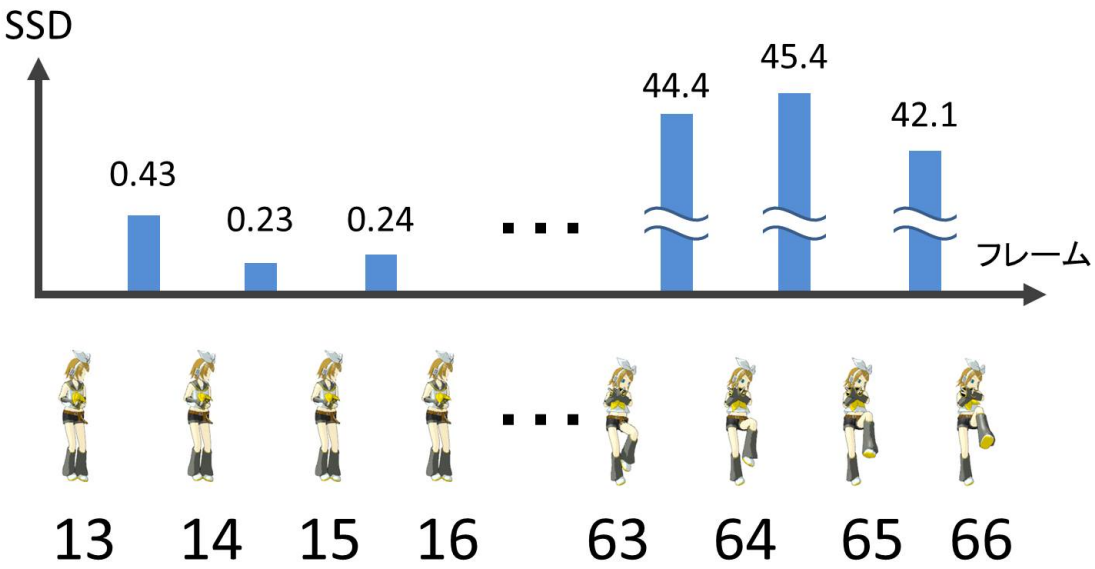


図 5.3　各フレーム間の SSD のグラフと姿勢。縦軸はフレーム間のジョイントの座標の二乗距離の総和 (SSD) を示す。横軸はフレームを示す。64 フレーム目と 65 フレーム目の間の移動量が最も大きいので 65 フレーム目が中無しによって抜かれる。また、14 フレーム目と 15 フレーム目の移動量が小さいので、このフレーム間に 14 フレーム目のデータがコピーされる。

Algorithm 2 中無しによる速さの強調

$N_{io} \leftarrow$ user-specified number of inbetween omission
for $i \leftarrow 1$ to N_{io} **do**
  $k_{max} \leftarrow \mathrm{argmax}\, SSD(k)$
  delete k_{max}-th frame
  $k_{min} \leftarrow \mathrm{argmin}\, SSD(k)$ s.t. $\#\textsc{Duplicated}(k-1) < 3$
  duplicate $(k_{min}-1)$-th frame
end for

5.2.3　コマ撮りによる細かい動きの低減

次にコマ撮りの処理を再現することで、微細な動きの低減を行う (図 5.4)。SSD が最小となるような k_{min} 番目のフレームを求め、k_{min} 番目のフレームに $(k_{min}-1)$ 番目のフレームを上書きすることで実現する。この処理を貪欲的に繰り返すことで、動きが少ないところの姿勢行列に変化がなくなり、細かい動きが低減される。同じフレームが続くと動きが完全に止まってしまうため、中無しと同様に同じフレームが連続できる数は 3 フレームまでとする。この処理の疑似コードを Algorithm 3 に示す。$\#\textsc{Copied}(k)$ は k 番目のフレームを後続のフレームに上書きした数を意味している。

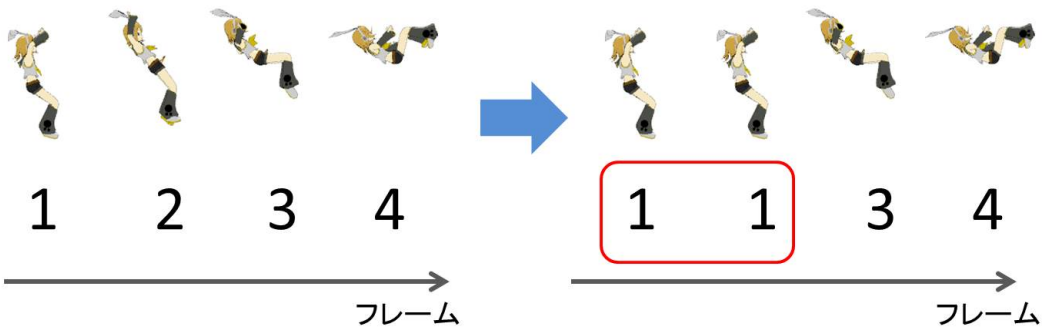


図 5.4　コマ撮り。SSD(1) が小さいため、2 フレーム目に 1 フレーム目をコピーする。こうすることで、細かい動きが低減される。

Algorithm 3 コマ撮りによる細かい動きの低減

$N_{hf} \leftarrow$ user-specified number of holding frames
for $i \leftarrow 1$ to N_{hf} **do**
  $k_{min} \leftarrow \mathrm{argmin}\, SSD(k)$ s.t. $\#\textsc{Copied}(k-1) < 3$
  copy $(k_{min}-1)$-th frame to k_{min}-th frame
end for

5.2.4　代替手法についての検討

中無しを行う際、本手法ではフレームを抜いた後に SSD を再計算し、再びフレームを抜いている。別の方法として、全フレームに対し SSD を計算して SSD が大きい上位のフレームを一度に抜くという手法も考えられるが、この手法では本手法に比べ元のモーションからの変化が少ない (図 5.5)。本手法では抜かれるフレームが連続するため、同じ箇所からフレームが抜かれ動きの速さが強調される。一方で、一度に抜く手法では抜かれるフレームが連続せずに様々な箇所から抜かれ、元のモーションとの変化が少なく、動きの速さが強調されにくい。

中無しとコマ撮りのどちらを先に行うかという順番に関して、本手法では中無しを行った後にコマ撮りを行っているが、この順番を逆にして実験を行ったところ大差が見られなかった。理由は、コマ撮りで抜かれるフレームは動きが小さい部分であり、中無しで抜かれる可能性があるフレームには互いに影響を及ぼさないからである。

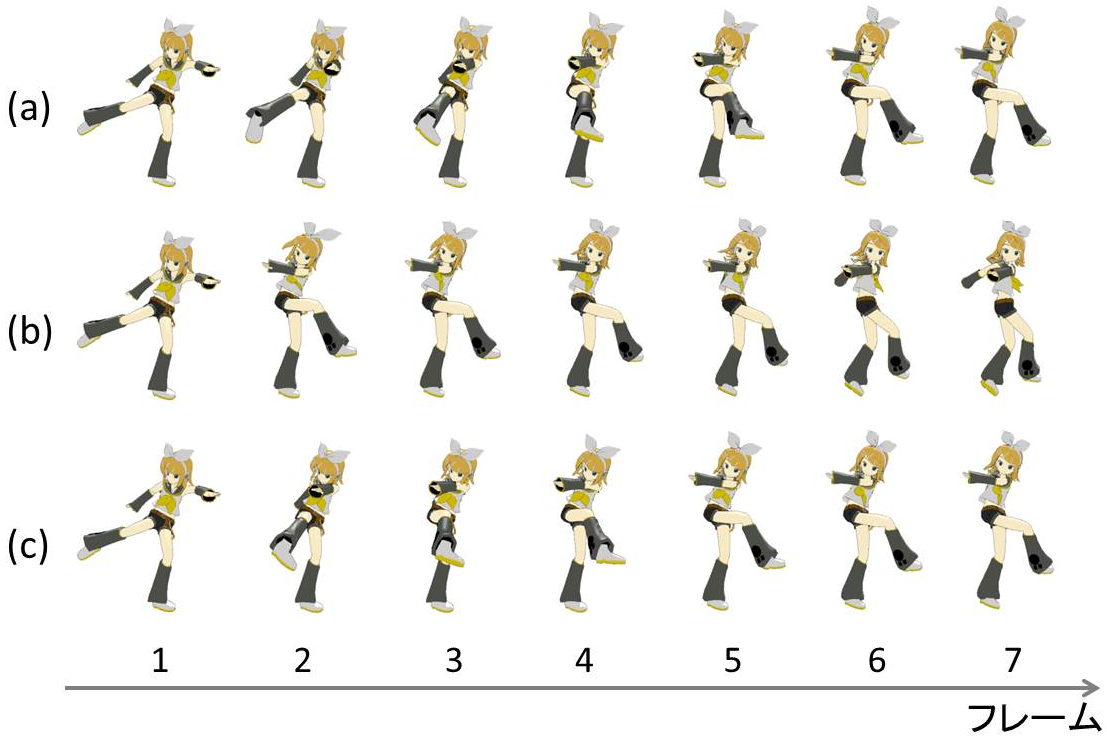


図 5.5　本手法 (b) とフレームを一度に抜く方法 (c) の比較。(a) は入力モーションデータ。本手法では 2 フレーム目から 5 フレーム目までが抜かれるが、フレームを一度に抜く方法では 2 フレーム目しか抜かれないため、繰り返し抜く方が動きの速さが強調されやすいことがわかる。

5.3　結果

本手法を C++ と Qt Library を用いて実装した。実行環境は Intel Core i7-2760QM 2.40GHz CPU の PC を用いて実行した。モーションキャプチャのデータは Mocap-data.com [4] と CMU Graphics Lab Motion Capture Database [1] で公開されているものを用いた。3D モデルは MikuMikuDance [3] に含まれているものを用いた[*1]。本手法を用いて得られた結果を図 5.6, 5.7, 5.8, 5.9 に示す。これらの図の上段は入力に用いたモーションキャプチャのデータであり、下段は本手法で得られた結果である。いずれも 24FPS で出力したものの一部を表示している。本手法の結果は動きの速さを強調する

[*1] 鏡音リンは、クリプトン・フューチャー・メディア株式会社の登録商標である。

ものであり、スナップショットではわかりにくいため、動画ファイルを参照されたい[*2]。図 5.6 では、上段の 3 フレーム目から 6 フレーム目が中無しによって削除されているため、キャラクタの右腕の動きの速さがはっきりと強調されている。同様に、図 5.7 は蹴る動作に本手法を適用したものである。上段の 2 フレーム目からフレームまでが削除され、右足の動作のスピード感が増していることが確認できた。図 5.8 はテニスのスウィングの動きに適用したものである。2 フレーム目から 5 フレーム目の動きが削除されていることで、振りの速さの強調が確認できた。図 5.9 は野球のスウィングの動きである。2 フレーム目から 7 フレーム目の動きが削除され、速さが強調されている。

図 5.10 は、図 5.6 と同じモーションの一部の区間のキーフレームを表示したものである。青い棒があるフレームはキーフレームであり、棒がないフレームはコマ撮りによって削除され、前フレームをコピーして挿入したフレームであることを示している。動きが大きいところはキーフレームが多く、少ないところはキーフレームが少なくなっている。これを動画としてみると微小な動きが低減されたことが確認できた。

表 5.1 は、それぞれのモーションのフレーム長、中無しで抜いた総フレーム数 N_{io}、コマ撮りでコピーした総フレーム数 N_{hf}、処理時間をまとめたものである。中無しの数 N_{io} とコマ撮りの数 N_{hf} に関しては、何枚抜けば適切かという枚数はアーティストの感覚に依存するため一意に決めることは難しく、本手法ではユーザが枚数を設定できるようにしている。我々が行った実験では、中無しの数 N_{io} は 4 フレームから 6 フレームくらいが適当であり、それ以上抜くと動きがとびすぎて一連の動作に見えなくなることがわかった。またコマ撮りによってコピーされるフレームの数 N_{hf} はフレーム長の 3 分の 1 くらいが適当であった。

表 5.1　実験結果

ファイル名	フレーム長	N_{io}	N_{hf}	処理時間 [ms]
Pitch	129	4	33	69
Kick	92	4	30	28
Tennis	88	4	29	27
Baseball swing	120	6	40	63

[*2] 動画は下記の URL にある。
`http://vacation.aid.design.kyushu-u.ac.jp/~maki/projects/limited_animation/limited_animation_ja.html`

図 5.6　Pitch。入力となるモーションデータ（上段）と本手法で得られた結果（下段）の一部のシーケンスを示す。フレーム長が 129 フレーム、中無しを 4 フレーム、コマ撮りを 69 フレーム適用。右腕の動きの速さが強調されている。

図 5.7　Kick。フレーム長が 92 フレーム、中無しを 4 フレーム、コマ撮りを 30 フレーム適用。足の動きの速さが強調されている。

図 5.8　Tennis。フレーム長が 88 フレーム、中無しを 4 フレーム、コマ撮りを 29 フレーム適用。右腕の動きの速さが強調されている。

図 5.9　Baseball swing。フレーム長が 120 フレーム、中無しを 6 フレーム、コマ撮りを 40 フレーム適用。両腕の振りの速さが強調されている。

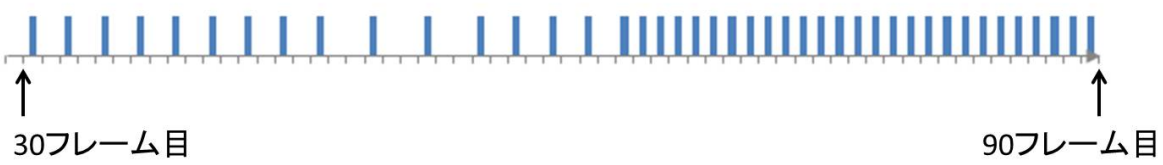


図 5.10　図 5.6 のモーションの一部の区間のキーフレーム。

5.4　考察

本章では疑似三次元モデルのキャラクタのためのアニメーションとしてカートゥーンアニメーションで用いられているリミテッドアニメーションというスタイルに着目し研究を行った。リミテッドアニメーションの表現手法のコマ撮りと中無しに着目し、モーションキャプチャのデータを入力としてそれらの表現の再現を試みた。キャラクタの動きのフレーム間での各々のジョイント位置の二乗距離の総和 (SSD) に着目し、SSD が大きいフレームを削除することで中無しの効果が再現でき、また、SSD が小さいところをコマ撮りにすることで、微小な動きが低減されたことを確認した。

本手法では SSD を計算する際、ワールド座標系でのジョイントの座標値で計算している。このためワールド座標系では SSD が大きくても、カメラとオブジェクトが離れているときはスクリーン座標系での SSD は小さいといったように、ワールド座標系での SSD とスクリーン座標系での SSD が一致しない場合もありうる。これに対応するためにスクリーン座標系での移動量も考慮する必要がある。

本手法では誇張のためのモデルの変形は行っていないが、リミテッドアニメーションの表現では変形も重要な要素である。本手法とカートゥーンアニメーションフィルタ [43] のような誇張表現を組み合わせることで、よりリミテッドアニメーションに近い表現ができると考えられる。

第 6 章

結論

本論文では CG における疑似三次元モデルの三次元モデリングおよびアニメーション手法に関する一連の研究をまとめた。疑似三次元モデルとは二次元モデルの基本構造を持ちながら、疑似的に奥行き情報を表現しているモデルであり、コンピュータ上で手描きの質感を持ったカートゥーンキャラクタや二次元的移動に限定された見下ろし型シーンの表現に使われる。本論文では疑似三次元モデルを三次元モデルとして操作できるようにし、三次元空間での疑似三次元モデルのための動きの生成を目的とし、次の 2 つの課題を解決した。

1. 疑似三次元モデルという二次元のデータから、三次元モデルを作る際の奥行きを推測する手法の開発。
2. 生成された三次元モデルに動きを付ける際、疑似三次元モデルに適した動きを生成する手法の開発。

1 つ目の課題に関して、本論文では疑似三次元モデルを、建物や地形などを表すシーンと人物や動物などを表すキャラクタ分け、それぞれにおいてこれらを三次元モデルとして扱うための手法の開発を行った。2 つ目の課題は疑似三次元モデルのキャラクタの三次元モデリングの後に、三次元空間で動きを付ける際に生じる課題であり、これを解決するために疑似三次元モデルのキャラクタのための動きの生成も行った。

6.1　本研究の成果

6.1.1　疑似三次元モデルを用いたシーンの三次元モデリング

コンピュータゲームで用いられている表現の見下ろし型視点表示という疑似三次元モデルに着目し、見下ろし型視点表示を三次元モデルとして操作できるようにするための手法の開発を行った。見下ろし型視点表示とは、真上からの見た目の情報と正面からの見た目の情報を同時に表示している表現手法である。この表現において水平方向の情報は地形の幅を表しており、垂直方向の情報は地形の高さまたは奥行きを表している。見下ろし型視点表示は従来の三次元モデルの表示とは異なる表現方法であるが、我々は見下ろし型視点表示からおおよその地形の形状を知覚できる。その性質を利用し、見下ろし型視点表示で三次元モデリングを行うためのインタフェースの開発を行った。また、見下ろし型視点表示は従来の三次元 CG で表示できないため、従来の三次元 CG での表示を可能にするための表現手法の開発も行った。

6.1.2　疑似三次元モデルを用いたキャラクタの三次元モデリング

疑似三次元モデルを用いたキャラクタに着目し研究を行った。手描きのイラストで描かれたキャラクタは誇張して表現されることがあるため、正面からの見た目と側面からの見た目に無理が生じることがある。このようなキャラクタを従来の三次元 CG で扱うことは困難である。本論文ではこのような表現を扱うことができる疑似三次元モデルのキャラクタに着目し、このような疑似三次元モデルを三次元モデルとして扱うための手法の開発を行った。具体的には、ビルボード構造によるキャラクタモデリングを行う。入力として異なる視点から描かれた 2 枚のキャラクタのイラストとそれが描かれた向きを用いる。初めに、イラストを目や口などの閉領域に分割し、イラスト間での閉領域の類似度に基づいて閉領域を対応付ける。対応付けられた閉領域での三次元空間での位置を推定し、その位置にビルボードとして各閉領域を配置する。そして視点に応じてビルボードの形状を変形させることで三次元モデルとしての操作を可能にした。また、閉領域の類似度の計算は既存手法を改良することで、より精度よく対応付けを行うことが可能になった。

6.1.3　疑似三次元モデルのためのアニメーションの生成

疑似三次元モデルのキャラクタのためのアニメーションの研究を行った。疑似三次元モデルを用いた表現の一つであるカートゥーンアニメーションではキャラクタの動きを誇張して表現することがある。この表現スタイルはリミテッドアニメーションと呼ばれる。リミテッドアニメーションとは、同じ絵を 2 コマあるいは 3 コマ連続して用いる「コマ撮り」によって、毎秒 24 コマのアニメーションを構成する表現手法である。元々はコスト削減のために開発されたが、意識的に絵や動きに制限を加えた表現としても発展し特有のメリハリのある表現を可能にした。一方、従来の三次元 CG でアニメーションを制作する場合は、実際の人間の動きをモーションキャプチャでキャプチャし、それを三次元モデルに適用して動きを生成する。しかし、モーションキャプチャの動きをそのまま疑似三次元モデルであるカートゥーン調のキャラクタに適用すると違和感が生じるため、本論文ではモーションキャプチャのデータをリミテッドアニメーション風にするための手法の開発を行った。具体的には、モーションキャプチャのデータを用い、キャラクタの動きが速いフレームを抜くことで動きの速さを強調し、キャラクタの動きが小さいフレームをその前フレームと同じ姿勢にすることで動きの微細な振動を低減するという手法を開発した。本手法により、モーションキャプチャから得たデータをリミテッドアニメーション風に表現できるようになり、疑似三次元モデルのキャラクタのためのアニメーションの作成を可能にした。

6.2　今後の展望と課題

本論文では疑似三次元モデルをシーンとキャラクタに分類し、それぞれを三次元モデルとして扱うための手法の開発を行った。疑似三次元モデルを用いたシーンの表現では、現在のところ角が丸いような地形を表現することができない。丸い形状の表現が可能になれば、より幅広いシーンの生成が期待できる。疑似三次元モデルを用いたキャラクタの生成に関しては、モデリングワークフローにおいて手作業が発生する箇所があり、それらの自動化を行うことで制作コストが下がると考えられる。具体的には、遮蔽されて見えない部分の形状の補完をする際に、遮蔽されていない別視点の領域の情報を用いれば自動で遮蔽領域の推定ができるのではないかと考えられる。疑似三次元モデルのキャラクタのためのアニメーションに関しては、キャラクタの動きの速さだけでなく、動いている際にもキャラクタの形状を変形して誇張表現を行うことでより豊かな表現ができると考えている。また、現在の仕組みでは疑似三次元モデルのシーンとキャラクタを同時に

扱うことは考慮されていない。今後はこれらの表現を一つの空間で扱うことができるような手法の開発が課題である。

謝辞

本論文の執筆にあたりご指導くださった九州大学大学院芸術工学研究院の鶴野玲治准教授に感謝いたします。また、本論文をまとめるにあたり貴重なご意見をいただきました同大学、藤村直美教授、富松潔教授に感謝いたします。本研究を推進するにあたっては、筑波大学大学院システム情報工学研究科の金森由博助教には様々なご指導をいただきました。深く感謝いたします。また、同大学の三谷純教授、福井幸男名誉教授にも様々なご意見をはじめ研究生活での様々なご支援をいただきました。ここに感謝いたします。そのほかにも、九州大学の博士課程での先輩である安東遼一氏には研究生活での多大なアドバイスをいただきました。ここに感謝いたします。最後に、筆者の研究を支えてくださった家族に感謝いたします。

参考文献

[1] *CMU Graphics Lab Motion Capture Database*. http://mocap.cs.cmu.edu/.

[2] *Esri CityEngine*. http://www.esrij.com/products/esri-cityengine/.

[3] *MikuMikuDance*. http://www.geocities.jp/higuchuu4/.

[4] *Mocapdata.com*. http://www.mocapdata.com/.

[5] *RPGVX Ace*. http://tkool.jp/products/rpgvxace/indexVX.

[6] ケータイでつづられる新たな『*FF*』――『*FF* レジェンズ』配信決定. http://dengekionline.com/elem/000/000/282/282799/.

[7] ファイナルファンタジー *III*. http://www.nintendo.co.jp/wii/vc/vc_ff3/index.html.

[8] *[*メイキング*]* サンジゲン *3D* について. http://www.noitamina-brs.jp/special/making3d.html.

[9] Ken-ichi Anjyo, Shuhei Wemler, and William Baxter. Tweakable light and shade for cartoon animation. In *Proceedings of the 4th International Symposium on Non-photorealistic Animation and Rendering*, NPAR '06, pages 133–139, New York, NY, USA, 2006. ACM.

[10] William Baxter, Pascal Barla, and Ken Anjyo. N-way morphing for 2d animation. *Comput. Animat. Virtual Worlds*, 20(2-3):79–87, June 2009.

[11] William Baxter, Pascal Barla, and Ken-ichi Anjyo. Rigid shape interpolation using normal equations. In *Proceedings of the 6th International Symposium on Non-photorealistic Animation and Rendering*, NPAR '08, pages 59–64, New York, NY, USA, 2008. ACM.

[12] W.V. Baxter, P. Barla, and K.-i. Anjyo. Compatible embedding for 2d shape animation. *Visualization and Computer Graphics, IEEE Transactions on*, 15(5):867–879, Sept 2009.

[13] Stephen Chenney, Mark Pingel, Rob Iverson, and Marcin Szymanski. Simulating cartoon style animation. In *Proceedings of the 2nd international symposium on*

Non-Photorealistic Animation and Rendering (NPAR), pages 133–138, 2002.

[14] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, pages 11–20, New York, NY, USA, 1996. ACM.

[15] Philippe Decaudin. Cartoon looking rendering of 3D scenes. Research Report 2919, INRIA, June 1996.

[16] F. Di Fiore, P. Schaeken, K. Elens, and F. Van Reeth. Automatic in-betweening in computer assisted animation by exploiting 2.5d modelling techniques. In *Computer Animation, 2001. The Fourteenth Conference on Computer Animation. Proceedings*, pages 192–200, 2001.

[17] Chie Furusawa, Tsukasa Fukusato, Narumi Okada, Tatsunori Hirai, and Shigeo Morishima. Quasi 3d rotation for hand-drawn characters. In *ACM SIGGRAPH 2014 Posters*, SIGGRAPH '14, pages 12:1–12:1, New York, NY, USA, 2014. ACM.

[18] Yotam Gingold, Takeo Igarashi, and Denis Zorin. Structured annotations for 2d-to-3d modeling. *ACM Trans. Graph.*, 28(5):148:1–148:9, December 2009.

[19] Alexander Hornung, Ellen Dekkers, and Leif Kobbelt. Character animation from 2d pictures and 3d motion data. *ACM Trans. Graph.*, 26(1), January 2007.

[20] Youichi Horry, Ken-Ichi Anjyo, and Kiyoshi Arai. Tour into the picture: Using a spidery mesh interface to make animation from a single image. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, pages 225–232, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[21] Eugene Hsu, Marco da Silva, and Jovan Popović. Guided time warping for motion editing. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, pages 45–52, 2007.

[22] Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. Teddy: A sketching interface for 3d freeform design. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pages 409–416, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[23] Takeo Igarashi, Tomer Moscovich, and John F. Hughes. As-rigid-as-possible shape manipulation. *ACM Trans. Graph.*, 24(3):1134–1141, July 2005.

[24] Satoshi Iizuka, Yoshihiro Kanamori, Jun Mitani, and Yukio Fukui. Efficiently modeling 3D scenes form a single image. *IEEE Computer Graphics and Applications*, 32(6):18–25, 2011.

[25] Takashi Ijiri, Shigeru Owada, Makoto Okabe, and Takeo Igarashi. Floral diagrams and inflorescences: Interactive flower modeling using botanical structural constraints. In *ACM SIGGRAPH 2006 Courses*, SIGGRAPH '06, New York, NY, USA, 2006. ACM.

[26] Ji-yong Kwon and In-Kwon Lee. An animation bilateral filter for slow-in and slow-out effects. *Graph. Models*, 73(5):141–150, 2011.

[27] Adam Lake, Carl Marshall, Mark Harris, and Marc Blackstein. Stylized rendering techniques for scalable real-time 3d animation. In *Proceedings of the 1st International Symposium on Non-photorealistic Animation and Rendering*, NPAR '00, pages 13–20, New York, NY, USA, 2000. ACM.

[28] John Lasseter. Tricks to animating characters with a computer. *SIGGRAPH Comput. Graph.*, 35(2):45–47, 2001.

[29] Yin Li, Michael Gleicher, Ying-Qing Xu, and Heung-Yeung Shum. Stylizing motion with drawings. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, pages 309–319, 2003.

[30] Xueting Liu, Xiangyu Mao, Xuan Yang, Linling Zhang, and Tien-Tsin Wong. Stereoscopizing cel animations. *ACM Transactions on Graphics (SIGGRAPH Asia 2013 issue)*, 32(6):223:1–223:10, November 2013.

[31] J. McCann, N. S. Pollard, and S. Srinivasa. Physics-based motion retiming. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, pages 205–214, 2006.

[32] Shigeo Morishima, Shigeru Kuriyama, Shinichi Kawamoto, Tadamichi Suzuki, Masaaki Taira, Tatsuo Yotsukura, and Satoshi Nakamura. Data-driven efficient production of cartoon character animation. In *SIGGRAPH 2007 sketches*, 2007.

[33] Byong Mok Oh, Max Chen, Julie Dorsey, and Frédo Durand. Image-based modeling and photo editing. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 433–442, New York, NY, USA, 2001. ACM.

[34] Junjun Pan and JianJ. Zhang. Sketch-based skeleton-driven 2d animation and motion capture. In Zhigeng Pan, AdrianDavid Cheok, and Wolfgang Müller, editors, *Transactions on Edutainment VI*, volume 6758 of *Lecture Notes in Com-*

puter Science, pages 164–181. Springer Berlin Heidelberg, 2011.

[35] Yoav I. H. Parish and Pascal Müller. Procedural modeling of cities. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 301–308, New York, NY, USA, 2001. ACM.

[36] Paul Rademacher. View-dependent geometry. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pages 439–446, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[37] Alec Rivers, Takeo Igarashi, and Frédo Durand. 2.5d cartoon models. *ACM Trans. Graph.*, 29(4):59:1–59:7, July 2010.

[38] Szymon Rusinkiewicz, Doug DeCarlo, and Adam Finkelstein. Line drawings from 3d models. In *ACM SIGGRAPH 2005 Courses*, SIGGRAPH '05, New York, NY, USA, 2005. ACM.

[39] Daniel Sýkora, Jan Buriánek, and Jiří Žára. Sketching cartoons by example. In *Proceedings of Eurographics Workshop on Sketch-Based Interfaces and Modeling*, pages 27–34, 2005.

[40] Daniel Sýkora, John Dingliana, and Steven Collins. As-rigid-as-possible image registration for hand-drawn cartoon animations. In *Proceedings of International Symposium on Non-photorealistic Animation and Rendering*, pages 25–33, 2009.

[41] Hideki Todo, Ken-ichi Anjyo, William Baxter, and Takeo Igarashi. Locally controllable stylized shading. *ACM Trans. Graph.*, 26(3), July 2007.

[42] Carlos A. Vanegas, Ignacio Garcia-Dorado, Daniel G. Aliaga, Bedrich Benes, and Paul Waddell. Inverse design of urban procedural models. *ACM Trans. Graph.*, 31(6):168:1–168:11, November 2012.

[43] Jue Wang, Steven M. Drucker, Maneesh Agrawala, and Michael F. Cohen. The cartoon animation filter. *ACM Trans. Graph.*, 25(3):1169–1173, 2006.

[44] Nayuko Watanabe and Takeo Igarashi. A sketching interface for terrain modeling. In *ACM SIGGRAPH 2004 Posters*, SIGGRAPH '04, pages 73–, New York, NY, USA, 2004. ACM.

[45] B. Whited, G. Noris, M. Simmons, R. Sumner, M. Gross, and J. Rossignac. Betweenit: An interactive tool for tight inbetweening. *Comput. Graphics Forum (Proc. Eurographics)*, 29(2):605–614, 2010.

[46] George Wolberg. Image morphing: a survey. *The Visual Computer*, 14(8-9):360–372, 1998.

[47] Jianxiong Xiao, Tian Fang, Ping Tan, Peng Zhao, Eyal Ofek, and Long Quan. Image-based façade modeling. *ACM Trans. Graph.*, 27(5):161:1–161:10, December 2008.

[48] Chih-Kuo Yeh, Peng Song, Peng-Yen Lin, Chi-Wing Fu, Chao-Hung Lin, and Tong-Yee Lee. Double-sided 2.5d graphics. *IEEE Transactions on Visualization and Computer Graphics*, 19(2):225–235, February 2013.

[49] コンピュータグラフィックス編集委員会. **コンピュータグラフィックス**. 財団法人画像情報教育振興協会（CG-ARTS 協会）, 2004.

[50] 尾沢 直志. **アニメ作画の仕組み**. ワークスコーポレーション, 2004.

[51] 藤子・F・不二雄. **ドラえもん /夏が来た！暑く熱く大冒険！！編/**. 株式会社　小学館, 2014.