2014年度　卒　業　論　文

リアルタイム 3DCG における
背景色を考慮した輪郭線描画手法の研究

指導教員：渡辺 大地 講師

メディア学部 ゲームサイエンス プロジェクト
学籍番号　M0111294
辻　圭佑

2014 年度　　卒　業　論　文　概　要

論文題目

リアルタイム 3DCG における
背景色を考慮した輪郭線描画手法の研究

メディア学部 **学籍番号：M0111294**	**氏名**	辻　圭佑	**指導教員**	渡辺 大地 講師

キーワード	3DCG、ノンフォトリアリスティックレンダリング、輪郭線、視認性、 リアルタイム、画像処理

近年 3DCG の技術の進歩で、ゲームグラフィックスや 3DCG を用いたアニメはより高品質なものに変化してきた。3DCG の研究分野には、2D のアニメーションや漫画のような表現を行うトゥーンレンダリングというものがある。その表現の 1 つに、オブジェクトの形状をよりわかりやすく表現するため、輪郭線を用いた誇張表現が存在している。従来の単色の輪郭線の描画手法では、物体と背景の色によって物体の輪郭が見えづらくなり、形状の視認性が低下する問題がある。これを回避するためには、より視認性の高い輪郭線にする必要があるが、オブジェクトの色とかけ離れた色はオブジェクトに対して違和感がある。また、単色の輪郭線では一部オブジェクト全体に対して視認性を高めつつ違和感の少ない色を設定するのは困難であり、部分的な色の変更が求められる。3DCG アニメーションでは輪郭線の色を部分的に変更している例がある。アニメーションの手法は視点やオブジェクトの位置が予め決まっているため、キャラクターと背景に対して輪郭線の色を適切に変更する事ができる。しかしゲームのようにリアルタイムグラフィックスでキャラクターやオブジェクトの移動に対応して描画していく場合には想定されるパターンが多く、全てに対して輪郭線の色の変更を行うのはコストが掛かる。本手法ではオブジェクトの持つ輪郭線の色に着目し、リアルタイムで行える、物体の持つ形状の視認性を向上する手法を提案する。本手法ではオフスクリーンレンダリングによって予めキャラクターのみを描画した結果と、キャラクターと背景を合わせて描画した結果を出力する。その後、出力した 2 つの描画結果を比較し、キャラクターと背景の境目である画素を検出する。画素を検出した場合、その画素を輪郭線として決定し描画する。輪郭線の最終的な描画の前にはキャラクターと背景の色の差を比較し、ユーザーの入力した値よりも画素の持つ色の色相・彩度が近く、境目の認識が難しいと判断した場合にその輪郭線の色をより明るく、もしくは暗くすることで見えやすい色の輪郭線を描画する。変更する際の元の色は、背景に隣接するキャラクター外縁の画素情報を加減して行うため、キャラクターの各部位とかけ離れた輪郭線が描画されるような結果になることを回避している。本研究の手法を用いた結果、リアルタイムの処理で輪郭線の部分に応じてより形状の視認性を向上しつつ違和感の少ない色を設定することが可能になった。

目 次

図 目 次

表 目 次

第 1 章

はじめに

近年、リアルタイム 3DCG の分野は急速に発達してきており、様々な表現の研究が行われている。

2D イラストの表現や絵画、水墨画等の手法を 3DCG の表現に取り入れた、ノンフォトリアリスティックレンダリング (Non-Photorealistic Rendering:以下 NPR)[1][2][3] も、3DCG という分野で盛んに行われている研究分野の 1 つである。大木 [4] は実写画像から 2D 風の背景画像を生成する研究を行った。2D アニメーションの表現を模した、セルルック、セルアニメ調の表現を行うトゥーンレンダリングという分野がある。トゥーンレンダリングではセルアニメーション特有の表現をグラデーションではなくきっかりとした階調を持つ陰影や、輪郭線によって表現する。安城 [5] はアニメ特有のハイライト表現を行った。

トゥーンレンダリングにおける表現の研究では、陰影の表現だけでなく、輪郭線による誇張表現も存在している。輪郭線は漫画やイラストで用いられる線画・主線の部分にあたり、本来の 3DCG モデルには存在しない要素であるが、2D らしさを表現するために 3DCG の表現でも用いられる手法である。近年では 3DCG を用いるゲーム作品やアニメーション作品が増加してきており、画面を構成する全てを 3DCG で作成する場合もあるが、2D グラフィックの背景の上で 3DCG のキャラクターを動かす場合もある。それ故に、2D の背景上でキャラクターが浮かないようにする 2D により近い表現には需要がある。『GUILTY GEAR Xrd -SIGN-』

[6] は 3DCG モデルで開発された 2D 対戦格闘ゲームである。『GRAVITY DAZE』[7] は背面法を使用して輪郭線を描画したモデルを使用した。榊 [8] は 3DCG のモデルに輪郭を付けることによる効果について述べている。松尾 [9] はオブジェクトの曲率によって輪郭線の太さを変更する事による誇張表現を行っている。川岸ら [10][11][12] は、輪郭の形を大きく変更することによって、アニメ的なブラー表現を再現した。地神 [13] は、輪郭線の抱える問題に LOD（Level of Detial）という、距離によって使用するモデルのサイズを変更する考えを用いることで改善した。このように、輪郭線を用いた誇張表現の研究がある。輪郭線は元々手書きの漫画、イラスト、絵画等の文化から出来たものであるため、それをデザインする人の筆圧や筆致など、人間特有の強弱や形がある。NPR の分野ではそういった強弱の再現や、輪郭線の変形によるデフォルメした誇張表現に対するアプローチが多い。また近年のアニメーション作品では、輪郭線の色に対してアプローチしている作品も存在する。『キャプテン・アース』[14] のエンディング映像では、キャラクターと背景の描画に使用されている線を全て赤紫色で描画することによって画面の与える印象に独特の統一感を持たせている。また、『二ノ国』[15] では、雪景色の背景に対して、キャラクターの輪郭線を灰色に変更することによって背景に馴染ませる表現を行った。これらの手法のように、近年のアニメやゲームでは、輪郭線の色を変更する手法が有る。

本研究では輪郭線の色について着目した。従来の手法の輪郭線は単色で描画される事が多く、これにより、キャラクターの 1 部の色と隣接する背景、それの境界となる輪郭線の色が近くなる場合、キャラクターの形状が視認しづらくなる現象が起きる。図 1.1 で、キャラクターの輪郭線が視認しづらくなる例を示す。

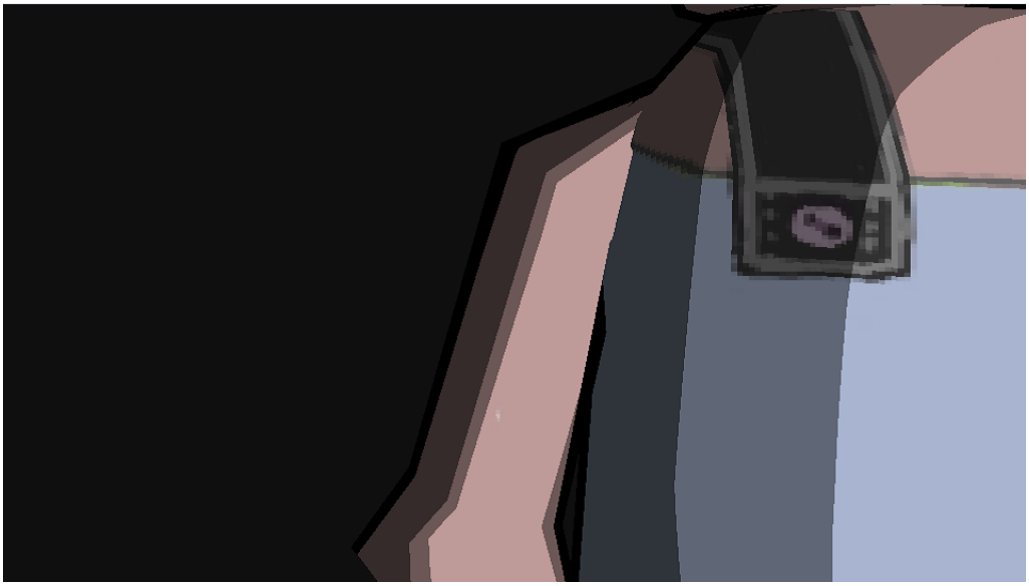

図 1.1: 視認しづらい輪郭線

輪郭線が視認しづらいという問題は、オブジェクトに対して目立つ色を設定すれば解決できる。次の図 1.2 は目立つ輪郭線の例である。このように目立つ輪郭線はオブジェクトに対して違和感が強く出てしまう。

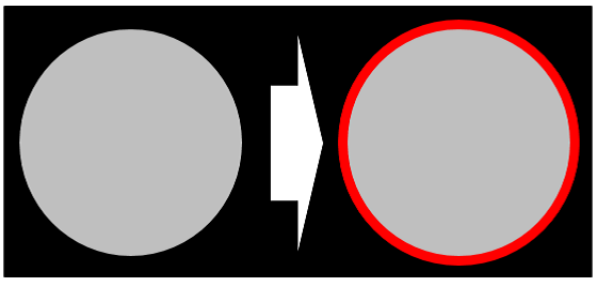

図 1.2: 目立つ輪郭線

『蒼き鋼のアルペジオ』[16] ではキャラクターとその背景で、キャラクターの影になっている部分が背景の暗い色に隣接している部分の輪郭線のみを明るい色へと変更している。これによって輪郭の存在している部分がはっきりとし、キャラクターの形状の視認性を向上した。部分的な色の変更のため、輪郭線の違和感も少ない。

このようにオブジェクトへの違和感を減らしつつ、単色の輪郭線によってオブジェクトの形状が視認しづらくなる問題を解決するためには、オブジェクトの色に対応して輪郭線の色を部分ごとに逐一変更していく方法が効果的である。この手法は3DCGのアニメーション作品で用いられた手法である。3DCGのアニメーションでは、予め撮影の際にカメラの位置が判明しているため、キャラクターに対してどのような背景が隣接するのかを確認し、その時に適した輪郭線の色へと変更を行うことが可能である。しかしゲーム中の操作キャラクターのように、動的に変化するものに対してグラフィックの変化をつけていく場合は、3DCGアニメーションで行うようにレンダリング結果を確認してから修正作業を行うという手法は用いることができず、リアルタイムレンダリングで処理結果を描画していかなければならない。また、ゲームのグラフィックではリアルタイムで処理できない重さの描画に関しては、予めテクスチャなどに書き込んで設定しておく方法が存在するが、オープンフィールドのゲームなどキャラクターの移動できる範囲が広いゲームほど、キャラクターの一部が視認しづらくなる場面は様々なパターンが想定できる。その全てを把握して、予め対応した輪郭線を設定しておくのは非常に困難である。よって、ゲーム上のリアルタイムグラフィックスで部分的な輪郭線の色変更を行うのは、より自動化した処理であることが望まれる。

以上のような問題を解決するために、本研究ではオブジェクトが持つ輪郭線をオブジェクトの外縁の画素とそれに隣接する背景の画素の色情報を比較し、その比較結果によって輪郭線画素の色の変更・決定を行う手法を提案する。本研究の目的は、対象のオブジェクトの視認性を向上することである。それに加えて、処理をリアルタイムで行えるようにすることによって、ユーザーが逐一輪郭線に対して変更処理を行うような作業コストを削減することも目的としている。また、オブジェクトの色を取得し、そこから輪郭線を出力することで、視認性を向上しつつ違和感の少ない輪郭線を描画することも目的である。本研究では、オブジェクトの存在している画面を1枚の2次元テクスチャとして扱うことで画像処理を行い、リアルタイムで輪郭線の色を部分的に変更する手法を用いた。その結果、リ

アルタイムでの処理が可能であり、尚且つより視認性の向上した輪郭線が描画できていることを確認した。

第2章では本研究の手法について説明し、第3章で本研究の手法の評価を、第4章でまとめを述べる。

第 2 章

本研究の手法

第 1 章で述べた輪郭線という単語は、それぞれの文献での利用の仕方に準じていたが、本研究での輪郭線は、あるオブジェクトとそれ以外のオブジェクトの境界線を指す単語である。

本章は 4 節で構成する。2.1 節でオフラインレンダリングによる、マルチレンダーパスについて述べる。2.2 節で描画した用法の比較から対象となるキャラクターの位置の判別方法について述べる。2.3 節では判別した情報から輪郭の位置を特定し、その近接画素との色情報の比較について述べる。2.4 節では比較した情報から、条件分岐でユーザーによって設定した輪郭の色に変更する方法について述べる。

2.1 マルチレンダーパス

マルチレンダーパスでは、レンダーターゲットという出力先を複数使用する。レンダリングの処理は通常画面に表示する情報を計算し出力することであるが、オフスクリーンレンダリングという方法を用いると本来表示する画面ではなく事前に用意したレンダーターゲットの中に描画結果を書き込むことが可能である。本手法ではオフスクリーンレンダリングした複数の結果に対して画像処理を行い最終的な出力画面を描画する。オフスクリーンレンダリングには GLSL(OpenGL Shading Language) のレンダーバッファーオブジェクト（Render Buffer Object 以下 RBO）

とフレームバッファーオブジェクト（Frame Buffer Object 以下 FBO）を使用する。FBO は画面以外に描画できる領域を提供するシステムであり、RBO は基本的に FBO とともに生成するものである。FBO に RBO を関連付けすることで、RBO をレンダーターゲットとして扱い描画結果をテクスチャのように使用することができる。

RBO にオフスクリーンレンダリングを行った結果は、最終的に画面に表示する情報を計算する際に使用することができる。今回は 2 つの RBO から最終的な画面を出力する。第 1 の RBO に対してはキャラクターのみを描画する。その際、OpenGL で予め設定する背景色が存在するが、キャラクター以外の画素の情報は全て Alpha 値を 0 に設定し、レンダリングを行う。画素は RGBA のステータスを持っており、RGB は赤緑青の色の情報、Alpha 値は透明度を表す値である。次の図 2.1 はキャラクターのみを描画した画面である。透明になっている部分は本来見えないので、今回は白で塗りつぶしている。

図 2.1: キャラクターのみを描画した画面

第 2 の RBO には、キャラクターに加え背景を含めた結果を描画する。次の図 2.2 がその例である。背景の部分は赤で塗りつぶしている。

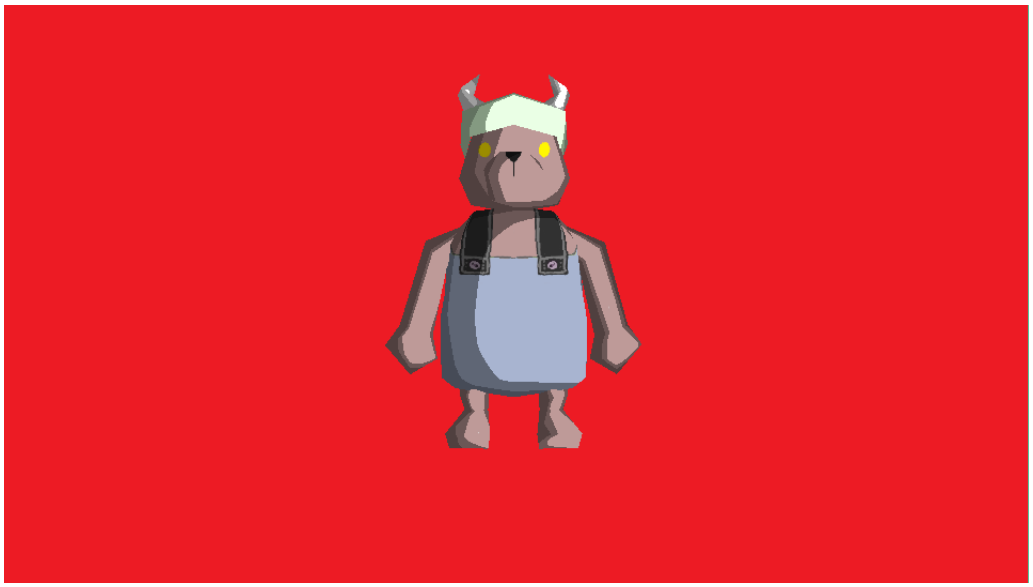

図 2.2: 背景とキャラクターを描画した画面

2.2　キャラクターの位置の判別

本手法では C++と OpenGL[17]、OpenGL 上でプログラマブルシェーダーとして用いる GLSL、そして FK[18] を使用した。GLSL の代表的なシェーダーには、頂点情報を扱う Vertex Shader（以下 VS）と頂点情報からモデルの面となる範囲を算出し、その範囲内の画素情報に処理を行う Fragment Shader（以下 FS）が存在する。今回は主に FS 上で処理を行う。

まず、2.1 節で得た 2 つの RBO に描画した情報を最終的な画面を出力する処理で 2 次元テクスチャとして読み込む。このテクスチャ2 枚の縦横の大きさは最終的に表示する画面の縦横の大きさと同じものであり、従って同じ縦横の画素数を持つ。画素は 2 次元テクスチャとして画面と同じ縦横の大きさを持つ矩形の中に配置しており、1 画素の縦横のサイズは、1 を縦と横それぞれの長さで割った値である。この 2 枚のテクスチャの画素を 1 画素ずつ同じ順番で踏査していき、2 つの RBO の画素情報を比較していく。1 枚目の RBO ではキャラクター以外の画素の Alpha 値を 0 に設定して描画しており、2 枚目の RBO ではキャラクターに加えて背景も一緒に描画している。画面上に描画するオブジェクトには基本的に 0 ではな

い Alpha 値を設定している。図 2.3 に画素情報の比較について示す。2 枚の RBO の画素データを比較した時、Alpha 値が一致するのはキャラクターを描画している画素だけである。そのため、Alpha 値の一致しない画素は背景を描画している画素であると判断する。

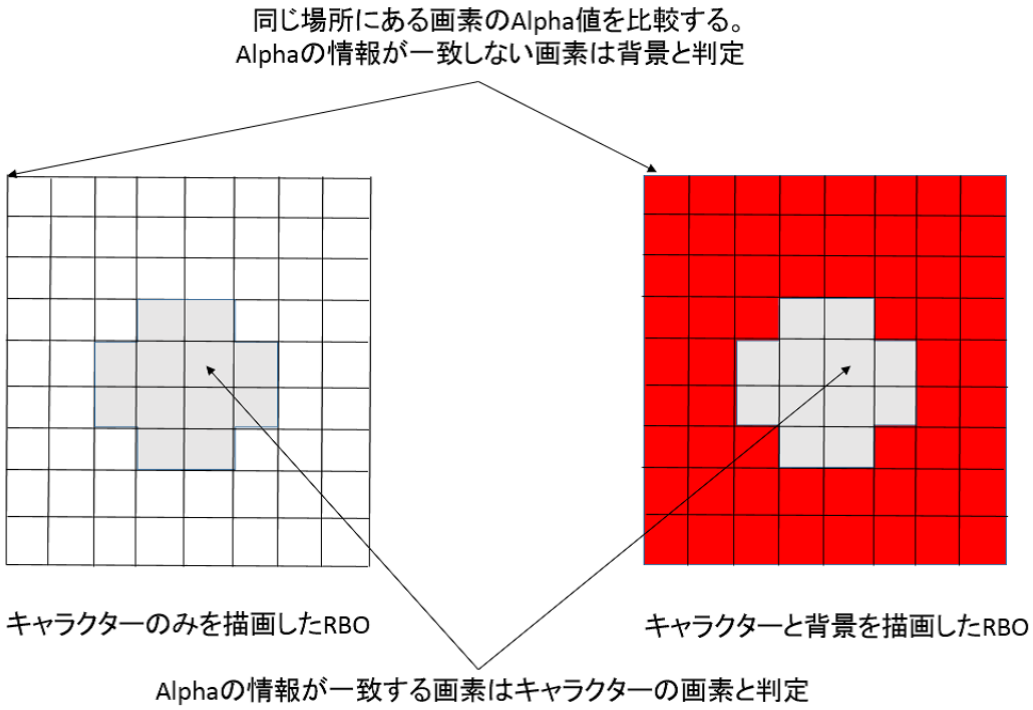

図 2.3: 画素情報比較

2.3 輪郭部分の判別

FS での処理は、テクスチャの持つ格子状に並んだ画素情報 1 つずつに対して行う。FS では現在処理を行っている画素から左右上下、ナナメ 4 方向の、合計 8 つの画素に対して 2.2 節の判別方法を行い、キャラクター画素か背景画素かを判別する。図 2.4 は輪郭画素決定の際に参照する画素の範囲である。現在の画素が背景画素であり、その隣接するいずれかの対象画素がキャラクター画素であった場合、現在の画素を輪郭線の画素として決定する。

図 2.4: 輪郭線画素の決定

輪郭線となる画素が決定したあとは、現在の画素の持つ色情報と、隣接するキャラクターの画素の色情報を比較する。比較には、比較対象となる 2 つの画素の RGB 情報と、そこから計算する HSV 方式 [19] の明度と彩度の値を使用する。それらを比較した差が、ユーザーの入力した値以下である場合には輪郭の色を変更する処理へ進む。標準では黒を設定している。

次に明度と彩度の算出方法を述べる。式の中で P_r は画素情報の赤要素であり、同じく P_g は緑の、P_b は青の要素である。GLSL 上で赤緑青それぞれの値は 0 から 1 の範囲で扱う。計算式は”色相、彩度、明度の計算方法” [20] を参照した。

比較する際に用いる明度 V は次の式 (2.1) で求める。

$$V = \max\left(P_r, P_g, P_b\right). \tag{2.1}$$

彩度 S の値は次の式 (2.2) で求める。

$$S = \frac{\max\left(P_r, P_g, P_b\right) - \min\left(P_r, P_g, P_b\right)}{\max\left(P_r, P_g, P_b\right)}. \tag{2.2}$$

2.4 輪郭線の色情報変更

本説では、色の変更処理を行う。今回はユーザーの入力によって明度を変更できるようになっている。数値の変更処理は 2.3 節で RBO から受け取ってきたキャ

ラクターの画素のRGB値を加減することによって行う。本手法では、明度を変更する情報は背景と接しているキャラクターの画素情報を元に行う。P_rはキャラクター画素の赤要素であり、同じくP_gは緑の、P_bは青の要素であり、ユーザーの入力をIとする。輪郭線の最終的な赤要素は隣接するキャラクター画素P_rの結果を全て足しあわせ、明度を上げる場合はその結果にIで乗算し、下げる場合はIで除算し、その結果を隣接するキャラクター画素の個数で割ったものである。同様の処理を緑要素と青要素に対しても行う。

明度はその画素のRGBのうち最大値を使用するが、最大値のみを変更した場合RGBどれかの色が目立ってしまうので、各色の比率をある程度維持するためにRGBそれぞれに同じ数値を乗算・除算する。この処理は、2.3節でも述べたとおりキャラクターがその時接している背景の画素情報から輪郭線を決めている。これによって、カメラが回転やキャラクターの移動によって背景と隣接するキャラクターの部位が変わった場合に、キャラクターの部位に対応した色を出力する。これは予めキャラクターの輪郭線に色を設定しておく手法では対応しきれない部分である。

第 3 章

検証と評価

この章では第 2 章で述べた手法が目的を達成しているかどうかについて検証し、評価する。検証を行う環境については次の表 3.1 に記す。

表 3.1: 検証を行う環境

OS	Windows8
CPU	Intel(R) Core(TM) i7-3632QM CPU @ 2.20GHz
GPU	Intel(R) HD Graphics 4000

3.1 本研究の効果の検証

本手法を用いた輪郭線の表現について検証する。既存の手法を適応したものが図 3.1、本研究の手法を適応したものが図 3.2 である。

図 3.1: 既存手法を適用したオブジェクト

図 3.2: 本研究手法で描画したオブジェクト

既存の手法は、背景とキャラクターの影になっている部分の色合いが近く、輪郭線が存在していることがわかりづらい。またオブジェクトの形状自体も把握しづらいことがわかる。対して、本手法を適応したモデルは色合いが近い部分の輪郭部分の色を変更しているため、輪郭部分がはっきりとし、より形状を視認しやすくなっていることがわかる。

図 3.3 は二色の背景に本研究手法を用いたオブジェクトを描画したものである。このように背景の色合いが大きく違う場合であっても正しく判定を行っている。

図 3.3: 二色の背景

図 3.4 と図 3.5 は図 3.3 を部分的に拡大したものである。

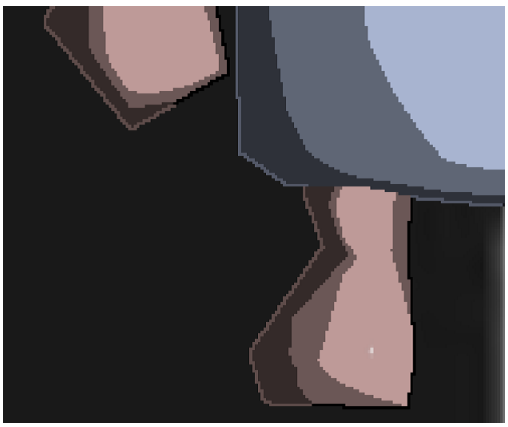

図 3.4: 拡大した図

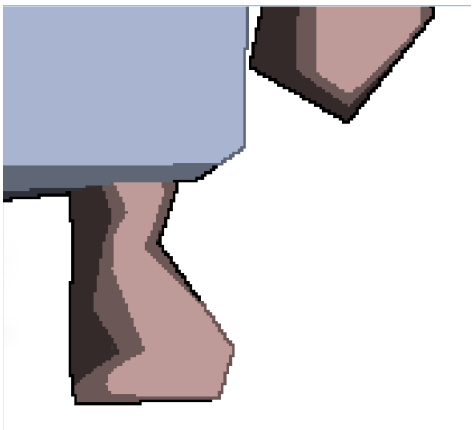

図 3.5: 拡大した図

リアルタイム 3DCG では、カメラの回転やキャラクターの移動によって、輪郭となる部分も常に変化することになる。図 3.6 はカメラの回転に対応した輪郭線を描画したオブジェクトである。カメラの回転などによって起こる角度の変更にも対応し、均一な輪郭が描画できている。

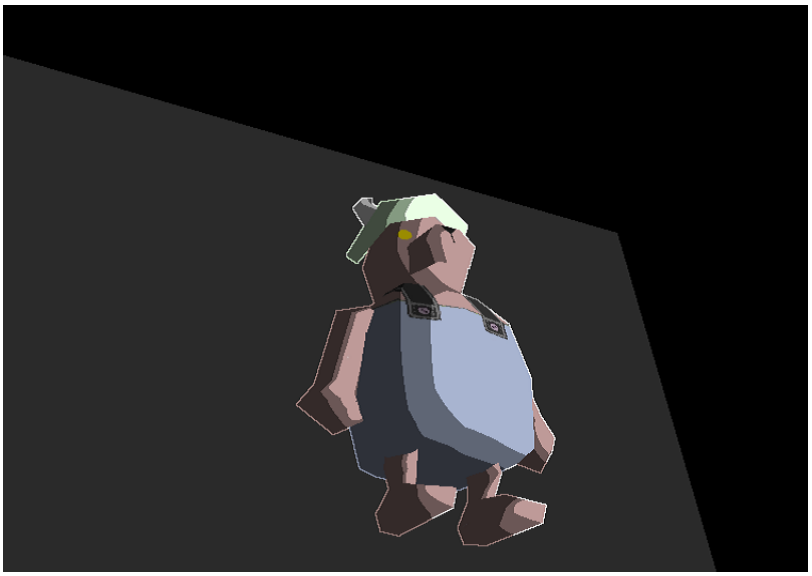

図 3.6: カメラの回転に対応

また、今回の手法を評価するにあたって、4 つのアンケート取り、検定を行った。アンケートの対象は大学生 16 名である。第 1 のアンケートは、本手法の輪郭線を用いてオブジェクトを描画した次の図 3.7 と、既存の黒単色の輪郭線を用いてオブジェクトを描画した図 3.8 でどちらの図の方がより輪郭が視認しやすいかアンケートを取り、カイ二乗検定を用いて検定した。その結果、$\chi^2 = 9, P = 0.0026 < .05$ で、本手法を用いた図に有意差が認められた。

図 3.7: 本手法を用いたオブジェクト

図 3.8: 既存の黒単色の輪郭線

第 2 のアンケートは同じように、2 色の背景で本研究手法を用いた描画結果 3.9 と、黒単色の輪郭線を用いて描画した結果 3.10 で、どちらがより輪郭が視認しやすいかアンケートを取った。結果は $\chi^2 = 4, P = 0.0026 < .05$ で、本手法を用いた図に有意差が認められた。

図 3.9: 本手法を用いたオブジェクト

図 3.10: 既存の黒単色の輪郭線

第 3 のアンケートは、単色の背景で本研究手法を用いた描画結果 3.11 と、赤単

色の輪郭線を用いて描画した結果 3.12 で、どちらの輪郭線がよりオブジェクトに違和感が少ないかアンケートを取った。結果は $\chi^2 = 12.254, P = 0.0046 < .05$ で、本手法を用いた図に有意差が認められた。

図 3.11: 本手法を用いたオブジェクト

図 3.12: 既存の赤単色の輪郭線

第 4 のアンケートは、2 色の背景で本研究手法を用いた描画結果 3.13 と、赤単色の輪郭線を用いて描画した結果 3.14 で、どちらの輪郭線がよりオブジェクトに違和感が少ないかアンケートを取った。結果は $\chi^2 = 16, P = 0.00063 < .05$ で、本手法を用いた図に有意差が認められた。

図 3.13: 本手法を用いたオブジェクト

図 3.14: 既存の赤単色の輪郭線

このことにより、本研究手法を用いた輪郭線が、既存の目立つ単色の輪郭線よりもオブジェクトに対して違和感の少ない輪郭線であると言える。

以上のことにより、本研究の手法は既存の手法で問題であったオブジェクトの輪郭線の視認性が下がる問題と、視認性を向上する際に輪郭線が目立つことによる違和感が大きい問題を解決した。

3.2 リアルタイム性の検証

本手法のリアルタイム性を検証する。検証には 3.1 で用いたものと同じ環境で行った。本手法では、キャラクターの移動やカメラの回転に対応した上で、動的な輪郭線の色変更が行えるかどうかについて検証する。検証では単色の背景を用意した環境でモデルを描画し、カメラの回転などを行った際の時間を計測する。モデルの描画にかかる時間は、Frame Per Second（以下 FPS）という単位で計測する。FPS とは、1 秒辺りに何回の描画処理が行ったのかを指すものであり、リアルタイム 3DCG のゲームでは、60FPS が標準である。今回はその 60FPS を 1 つの基準とし、1 体の時の FPS と 10 体の時の FPS を比較した。モデルのポリゴン数は

301 ポリゴンである。その結果が次の表 3.2 である。

表 3.2: FPS

オブジェクト総数	FPS
1 体	86.9fps
10 体	81.3fps

本手法では 10 体を描画した場合でも 81FPS を保てた。この数値は 60FPS よりも高い FPS である。このことから本研究の手法は、リアルタイム 3DCG を用いたゲーム上での適用も可能である。

第 4 章

まとめ

本研究にではリアルタイム 3DCG における輪郭線の表現について、オブジェクトと背景の色の配置によって視認性が低下する問題を改善する手法を提案し、評価を行った。その結果として以下のことが可能になった。まず、背景とオブジェクトの色相・彩度・明度の兼ね合いによって視認性が低下する現象の改善が可能になった。輪郭線の色に着目し、単色であることがほとんどであった輪郭線に対して、部分的な色の変更を行うことでオブジェクトの形状がより視認しやすくなった。オブジェクトの画素情報から輪郭線の色を決定することで、視認性を上げた際の違和感を減少できた。

加えて、リアルタイムでの輪郭線の色変更が可能になった。オブジェクトの存在する画面の情報をテクスチャとして扱い、画像処理を行うことで、リアルタイムでの処理に耐えうる輪郭線の色変更が行えるようになった。

残っている課題は、オブジェクトのパーツ毎の設定が行えないことである。本研究の手法では現状ではオブジェクト全体に同様の計算処理を行うことになる。したがって、キャラクターのオブジェクトで、髪の毛の輪郭線の色は変更せずに、服装の輪郭線に対してのみ本研究の手法を適用するなどの、細かい調整は行うことが出来ない。

本研究の今後の展望については、まず輪郭線の太さを変更することによる誇張表現との併用がある。現状の手法では、一定の品質の輪郭線を描画するために、全

体で均一な輪郭線を描画するアルゴリズムになっている。NPR の分野では、輪郭線の太さや形状を変えることによって行う誇張表現が研究されており、そういった手法との併用ができるようになれば、より表現できる幅が広がることになる。

謝辞

本研究を勧めるに際し、多くの指導を頂いた渡辺大地先生、三上浩司先生に心から感謝いたします。悩み事や躓いたことがあった時、相談に乗って頂いた院生の皆様、研究室メンバーにも同じく感謝いたします。皆様本当にありがとうございました。

参考文献

[1] Miln Magdics, Catherine Sauvaget, Rubn J. Garca, and Mateu Sbert. Post-processing npr effects for video games. *VRCAI '13 Proceedings of the 12th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry*, 2013.

[2] Stphane Grabli, Emmanuel Turquin, Frdo Durand, and Franois X. Sillion. Programmable rendering of line drawing from 3d scenes. *ACM Transactions on Graphics (TOG) Volume 29 Issue 2, March 2010 Article No. 18*, 2010.

[3] Martin Constable. Artist-led suggestions towards an approach in content aware 3d non-photorealistic rendering. *ACCV'10 Proceedings of the 2010 international conference on Computer vision - Volume part II Pages 142-151*, 2010.

[4] 大木花菜. 実写画像からの 2d アニメ用背景画像生成. 情報処理学会第 75 回全国大会, 2013.

[5] 安城健一. ハイライトシェーダー. *OLM TECHNICAL NOTE-OLMNOTE2004-001 August31, 2004*, 2004.

[6] アークシステムワークス. ギルティ・ギア xrd -sign-, 2014.

[7] ソニー・コンピュータエンタテインメント. Gravity daze/重力的眩暈:上層への帰還において彼女の内宇宙に生じた摂動, 2012.

[8] 榊正宗. 検証：3d アニメに輪郭線は必要か？, 2009. 参照: 2015-02-06.

[9] 松尾隆志. リアルタイム 3dcg における物体の形状を考慮した輪郭線誇張手法の提案. 芸術科学界論文誌 Vol.10, No.4, pp. 251–262, 2011.

[10] 古野泰. 3dcg における漫画的なスピード誇張表現に関する研究. 東京工科大学メディア学部 2006 年度卒業論文, 2006.

[11] 渡辺那. 3dcg 用いた背景制作における手描きアニメ調ブラーの表現. 東京工科大学メディア学部 2011 年度卒業論文, 2011.

[12] 川岸裕也. カートゥーンブラー:セルアニメーションのための非実写的モーションブラー. 情報処理学会研究報告, 2002.

[13] 地神知哉. 3dcg におけるキャラクターの表示サイズによる漫画的簡略化表現手法. 東京工科大学メディア学部 2007 年度卒業論文, 2007.

[14] キャプテン・アース製作委員会 MBS. キャプテンアース, 2014.

[15] 久代忠史, 侍絽吏玄蕃, 西川善司, 宮田悠輔, 永岡聡, 小村仁美. ゲームグラフィックス 2012. ワークスコーポレーション, 2012.

[16] サンジゲン. アニメ:蒼き鋼のアルペジオ -アルス・ノヴァ-, 2013.

[17] Mark Segal and Kurt Akeley. OpenGL4.0 グラフィックスシステムズ. 株式会社カットシステム, 273p 295p, 2010.

[18] 渡辺大地. *Fine Kernel Project*. http://fktoolkit.sourceforge.jp/. 参照: 2015-02-06.

[19] ISHIHARA WATARU. Hsv 方式. 参照: 2015-02-06.

[20] 画像処理ソリューション. 色相、彩度、明度の計算方法, 2013. 参照: 2015-02-06.