Vol.2018-CG-169 No.1
2018/3/3

Coarse Pixel Shading を用いた VR 酔い対策ための描画の高速化技法

橘 康貴[1]　今給黎 隆[1]

概要：Coarse Pixel Shading を用いた VR コンテンツの描画の高速化手法を提案する．近年，HMD デバイスの普及に伴い，VR コンテンツが増加してきている．HMD デバイスを用いた VR コンテンツでは，画面の更新頻度が低いと，画面の更新とユーザーが予測している動きの差異を感じやすく，不快感を催す VR 酔いが生じやすくなることが知られている．この VR 酔いを押さえる対策は VR コンテンツに重要な要素である．以前から，周辺視野の領域をユーザーから隠したり，モーションブラーを用いることで，VR 酔いを起きにくくする手法が提案されている．しかしながら，これらの手法は臨場感を損なったり，リフレッシュレートの低下を引き起こしていた．本論文では，照明計算の処理数の削減と，削減に伴う品質低下を補う手法を提案することにより，高速で臨場感を損なわない VR 酔い対策を実現する．本提案手法の効果は，描画性能を画面の更新速度の計測により検証を行い，VR 酔い対策の効果についてユーザーアンケートにより評価を行う．

A Fast rendering technique for VR sickness prevention using Coarse Pixel Shading

Koki Tachibana[1]　Takashi Imagire[1]

1. はじめに

近年，商業ベースの HMD の普及により，VR コンテンツコンテンツの数が増え，ユーザーが VR コンテンツに触れる機会が増えてきている．しかしながら，HMD のコンテンツでは，VR 酔い [1] と呼ばれる頭痛や吐き気を催す問題を生じやすいことが知られている．VR 酔いの要因として，見ている映像と自分の動きが一致していない事によるベクション (視覚誘導性自己移動感覚) が挙げられる．ベクション対策の一つとして，視野周辺に黒いマスクを掛けることで酔いを軽減する技術や，中心視野以外にブラーを掛けることで違和感を減ずる方法が提案されている．しかしながら，これらの方法では，視野が狭くなることで没入感が下がったり，ブラー処理の追加により，画面のフレームレートが低下するなどの問題点が生じ，没入感を損なわない高速な描画手法は確立しているとは言えない．

[1] 東京工芸大学
Tokyo Polytechnic University

本研究では，視野周辺の解像度を低下させるレンダリングを行うことで描画負荷を下げ VR 酔いを生じない描画方法を提案する．解像度の制御には，Coarse Pixel Shading (以下 CPS) を利用する．CPS は複数の画素を同じ色として出力し，ブロック状のアーティファクトが現れやすいため，品質向上のための低コストな改善手法を提案する．

提案法の評価としては，複数の手法に関する，実行時のフレームレートの計測を行い，VR 酔いの生じやすさ及び没入感に関してはアンケートを用いた評価を行う．

2. 関連研究

2.1 VR 描画の高速化

自然な VR を体感するには，リフレッシュレートの高速化が重要となる．Guenter ら [2] は，画面における視線からの距離に応じて解像度の異なる複数の画像を組み合わせる手法による視線追跡型のレンダリング技法を提案した．この手法では，同じ場所を複数回描画する必要がある．

描画する領域について注視する部分を判定し，部分的に

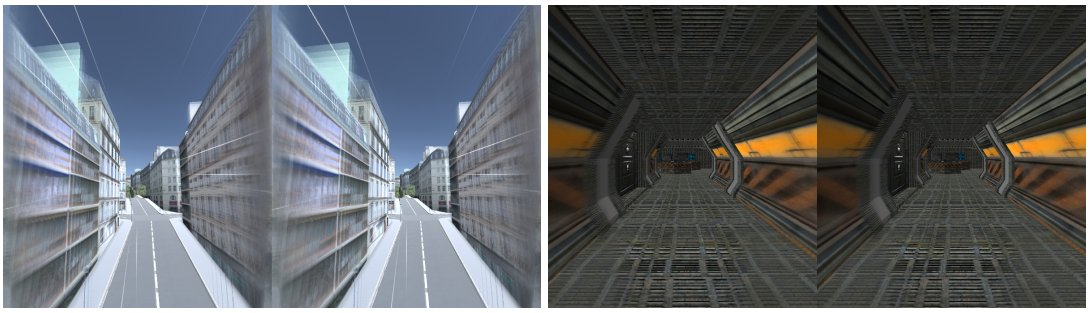

図 1　提案法による結果画像．視野周辺において詳細な情報が省かれている．

解像度を変更することで，高速に視線追跡型レンダリングを実現する手法も提案されている [3], [4]．Michael ら [5] は，視線を含めた知覚情報に基づき，画面内で解像度を変更してレンダリングする手法に関して，GPU を用いた高速なアルゴリズムを提案している．しかしながら，これらの手法は，一般的なゲームエンジンのレンダリングパイプラインで効率的に実行することは難しい．

実践的な VR の高速化手法としては，縮小画像による複数の解像度の画像を用意し，GPU の機能を用いて視線追跡型レンダリングを実現する手法が提案されている [6]．また，画面を複数に分割し，分割された画面ごとに解像度を調整する手法も提案されている [7], [8]．James [9] は，GPU の処理が 2×2 のフラグメントごとに計算されることから，このサイズを基本単位として，注視点からの画面での距離に応じて照明計算を行うフラグメントを間引くことで VR 映像を高速化する手法を提案した．これらの手法は，注視点を中心とした処理を対象としており，注視視野以外の周辺物体等の動きは考慮されていない．

2.2　VR 酔い

ユービーアイソフト株式会社の「イーグルフライト」[10] は，パリ上空を高速で飛ぶゲームである．HMD による一人称視点の VR ゲームでは，自身による動きの加速度を実際の体が感じることがないため，視覚情報のみから運動を推測する必要があるが，目で見た結果と自身の体験による予測が一致しないベクションによる VR 酔いが生じやすい．イーグルフライトでは，旋回時や視界の近くを建物が流れる際に，周辺視野となる左側，右側，下側について，ゲームの状況に応じて黒いマスクで覆うベクション対策を施している．しかしながら，視野周辺に対して黒いマスクを掛けることで情報の一部が失われてしまうため，VR コンテンツの没入感は低下してしまう．

千葉ら [11] は，複数のフィルタリング方法を比較し，視野周辺に対してぼかしをかけることが没入感を損なうことなく VR 酔いを抑えることを確認し，効果が高かったフィルタを CHIBA マスクとして提案している．CHIBA マスクは，没入感を損なうことなく VR 酔いの対策が実現できるが，ぼかし処理は多くのサンプリングを必要とするため，描画負荷は高くなる．

2.3　Coarse Pixel Shading

Vaidyanathan ら [12] は，照明計算を行うフラグメントに関して，フラグメントと画素を一対一に対応させるのではなく，2 のべき乗のサイズの正方形を複数の画素へ出力することで，レンダリングを高速化する CPS を提案した．

Rahul と Tomasz [13] は，CPS を遅延レンダリングに適応し，現在の商用ゲームで標準的に用いられているゲームエンジンでの CPS(遅延 CPS) を実現した．ただし，遅延 CPS は，GPU の描画パイプラインの中で閉じてはおらず，汎用的な GPU 計算を行う Compute Shader で CPS を実装しているため，十分な性能が出ているとは言えない．しかしながら，遅延 CPS は現在の GPU 及びゲームエンジンで動作することと，拡張がしやすい．本研究は，遅延 CPS を実装の基礎として用いた．

Patney ら [14] は，彩度を保存する画像処理とテンポラルアンチエイリアシング [15], [16] を多重スケールで用いて，より効果的な視線追跡型レンダリングを実現している．本提案法では，時間軸の効果について，異なる補間法を用いる．

3.　提案法

3.1　VR 描画の高速化

CHIBA マスクでは，周辺視野に対してぼかしをかけることで没入感を損なわない VR 酔い対策を実現した．しかし，ぼかし処理は負荷が高く，60fps 以上を維持することが推奨されている VR コンテンツでは，より高速なレンダリングが求められる．本研究では，遅延 CPS を拡張し，VR 酔い対策を施す描画の高速化を実現する．

3.1.1　中心窩レンダリング

VR 酔いの原因の 1 つであるベクションは，周辺視野の動きに強く影響を受けることが知られている [17]．本手法では，イーグルフライトを参考に，視野周辺で高速に動く物体の描画の詳細を除去する．3 次元の仮想空間において，画面上を高速に動く物体とは，カメラの近くに存在する物

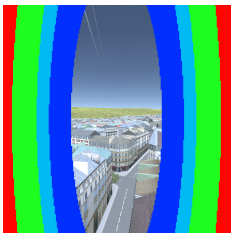

図 2　詳細度の可視化，塗りつぶされていない領域は最も詳細であり，青，水色，緑，赤と変わるにつれ詳細度を落とす．

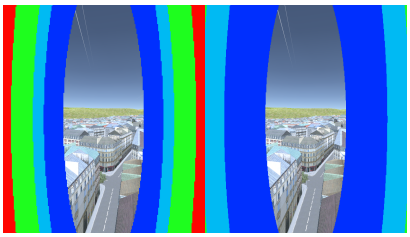

図 3　左右での詳細度の違い．左目の画面の方が，画面端に近づくにつれ，詳細度が大きく下がっている．

体である．そこで，視野周辺における物体の距離を測定し，物体が画面の近くであれば，ベクションが生じないように描画の詳細を低下させた描画を行う．具体的には，視点から左右と下方向の画面端へとレイを飛ばし，レイが衝突した物体までの距離に応じて，横方向と，下方向の描画の詳細度を決定する．画面の照明計算を行う際は，左右・下方向に関しての詳細度が低い場合，注視点からの角度が離れるに応じて描画する場所の詳細度を低下させる (図 2)．

低い詳細度の場所では，同じ色を複数の画素に割り当てる．詳細度に応じて，$1 \times 1, 2 \times 2, 4 \times 4, \ldots, 32 \times 32$ の大きさの画素を 1 つのフラグメントとして処理を行い，いずれかの場所の照明計算の結果を同じフラグメントに所属する全ての画素に記録する．

3.1.2　多重レベルディスパッチ

提案手法では，CPS を Compute Shader で実装を行う．これは，現在のピクセルシェーダで複数の画素に効率的に出力する仕組みがないことによる．しかしながら，Compute Shader で複数の画素に関する計算を行うのは効率が悪い．例えば，32×32 の大きさのブロックについて Compute Sheder で処理を行う際は，詳細度が低いことで 1 回の照明計算しか行わない場合に，1 つのスレッドの中で 1 回の照明計算しか行わないが，詳細度が最も高い場合には 1 つのスレッドで $32 \times 32 = 1024$ 個のフラグメントの照明計算が行われる．最悪の場合，1 つのスレッドが処理を終えた時間の 1000 倍の長さ待たされる可能性があり，照明計算の負荷が高い場合は，長い時間計算を行わないスレッドが発生する．提案手法では，この問題を解決するために，Compute Shader への投機を詳細度の段階の数に分ける．各投機で指定された詳細度のフラグメントだけの照明計算を行うことで，Compute Shader の空き時間が生じないレンダリングが実現できる．

3.2　アーティファクト除去

CPS で詳細度が低い際は，広い面積が同じ色で塗りつぶされるが，この効果は画面上に四角形のブロックの形状が目に見えるアーティファクトとして発生する．本研究では，このアーティファクトを消すための複数の処理を行う．

3.2.1　交互テンポラルフィルタリング

画素の色の平滑化の一つとして，時間軸の平均を行う．提案法では，一フレームおきに左右の目の画像について詳細度の強さを切り替えながら前のフレームの画像と合成を行う (図 3)．詳細度が低いフレームでは，レンダリングの速度を向上させ，詳細度が高いフレームでは，アーティファクトが少ない結果を生成する．合成する際は，詳細度が高い画像を強く合成することで，アーティファクトの低減の効果を高めることができる．

3.2.2　階層型バイリニアフィルタ

フラグメントの四角形の形状が目に見える要因は，フラグメントの境界で不連続に色が変わることにある．2 次元の軽量なフィルタリングを照明計算の後に用いた平滑化を導入する．照明計算された結果ついて，バイリニア補間などにより最近傍のピクセルとの平滑化を行うだけでは，詳細に表示されるべき領域はぼけてしまう．一方，平滑化は色が変化する部分でしか有効に働かず，フラグメントの大きさが大きいと四角形の形状がはっきりと残されてしまう．提案法では，フラグメントの大きさを考慮したバイリニアフィルタリングを定式化する．

照明計算した結果画像に関して，全画面を覆うフィルタリングを行う．照明計算の際に，各画素の色情報だけでなく，その画素の詳細度を出力する．今回の実装では，照明計算の結果の画像のアルファ成分に詳細度 s を格納した．フィルタリングを行う各画素 P でサンプリングされた色と詳細度情報 t_{00} 内に記録された詳細度 $s(P)$ から，照明計算の際のブロックの大きさ $L(P) = 2^{s(P)}$ 及び，ブロックの隅の位置 $O(P) = P \bmod L(P)$ を復元する．また，フィルタリングを行う画素の位置に関して，ブロックの位置と大きさを基に，縦方向と横方向について -0.5 から $+0.5$ の範囲で正規化されたブロック内での位置 $l(P)$ を計算する，

$$l(P) = (P - O(P))/L(P) - 0.5. \tag{1}$$

ここで，$L(p)$ の xy 成分に関して，その絶対値が大きな方を l_0，もう一方を l_1 とする (図 4)．P から l_0 方向に伸ばした隣接するフラグメント P_0 の情報を t_{10}，l_1 方向に伸ばした先のフラグメント P_1 の情報を t_{01} とする．通常のバイリニアフィルタリングが 4 点の情報を用いてサンプリングするのと同様に，P_0 に関して，l_0 に垂直な方向で隣接す

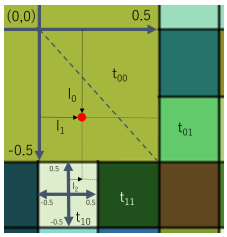

図 4　階層型線形補間のパラメータ．

るフラグメント P_2 を求め，その情報 t_{11} をサンプリングする．また，合成の係数の算出のため，P_0 における正規化されたブロック内での位置 $l(P_0)$ を計算し，その l_0 に垂直な成分を l_2 と置く．得られた色情報は，正規化されたブロック位置を用いて合成を行う．具体的には，隣接するフラグメントに関して，対応する方向の係数を重みの絶対値で線形補間することで最終的な色を生成する (図 5)，

$$output = lerp(lerp(t_{11}, t_{10}, ||l_2||), lerp(t_{01}, t_{00}, ||l_1||), ||l_0||), \quad (2)$$

$$lerp(A, B, t) = t\,A + (1-t)\,B, \quad (3)$$

ここで，$||\cdot||$ は，ベクトルの大きさである．

4. アンケート手法

Kennedy らの提案した，VR 酔いや疲労の分析に有効と考えられる Simulator Sickness Questionnaire(SSQ) [18] により VR コンテンツの体験前と体験後の体調の比較を行った．SSQ は，定められた 16 個の各項目に関して，0 から 3 の 4 段階でユーザーに評価してもらう主観的評価である．アンケート結果は，定められた手順により集計され，最終的に Nausea(気持ち悪さ), Oculumotor(眼精疲労), Disorientation(ふらつき感), TotalScore(総合点) という 4 つのスコアが導出される．

SSQ アンケートは VR 酔いについての評価であり，没入感が損なわれているかについて評価できない．本研究では，Aikawa の研究 [19] を参考に，「臨場感」，「スピード感」，「品質」，「周辺視野」，「ゲームルール」の各項目に関して，1 から 5 の 5 段階の尺度でユーザーからのアンケートを収集した．

アンケート調査は，千葉ら [11] の手法を参考に，ユーザーごとにゲームを 2 回体験してもらう形式で実施した．まずプレイ前に SSQ アンケートを行い，被験者の体調の調査を行う．次に，一回目のゲームプレイを行い，体験後に SSQ アンケートと品質アンケートを実施する．その後，二回目のゲームプレイを行い，その後で再び SSQ アンケートと品質アンケートを実施した．技法の比較のために，2 回のゲームプレイのどちらかでフィルタリングを行った．プレイの順番による差異を検証するために，被験者の半分は最初はフィルタリング行わず，2 回目でフィルタリングを行うプレイを実施し，残りの半分の被験者については，最初にフィルタリングを行うプレイを実施した．2 回のプレイのどちらがフィルタリングされているかは，被験者は知らされない．

5. 結果

本提案手法を，3 次元空間を自動的に進むフライトアクションゲームにより検証した (図 6)．実験には，CPU が Intel Corei7-7700HQ，GPU が GeForce GTX 1070 のノート PC を用いた．HMD には，視線追跡を行うことができる Fove 0 を利用した．

5.1 パフォーマンス

表 1 が，実行速度の測定結果である．フィルタリングを掛けていない状態の実行速度は，81.7 fps で，提案手法による同シーンは 26.3 fps となった．提案手法は，遅延 CPS を拡張して実装しており，Compute Shader で照明計算を行うため遅い結果となっている．本来の手法に対して，照明計算をピクセルシェーダではなく，Compute Shader に置き換えた際の実行速度は 17.0 fps であり，実行率としては 4.8 倍の開きがある．したがって，提案手法が十分に描画パイプラインで効率的に実行することができれば，この開きである約五倍の実行速度で動作することが見込まれ，その際は 126.2fps で動かせることが期待できる．

本提案手法と他のベクション対策についての比較を行う．イーグルフライトと同様に，詳細度が低くなるべき部分を暗くする処理では，同じシーンにおける実行速度は 38.0 fps となった．イーグルフライトの処理は，臨場感の低下を許しても高速に実行できることを目指していることが想定でき，本提案手法よりも高速な結果となっていることは自然である．同じシーンにおいてぼかしマスクと同様な詳細度が低い部分をぼかす処理を行った際は，79.2 fps となった．ぼかし処理を強くかけるには，より広い領域の画面情報を必要とするため，実行速度は低下する．

多重レベルディスパッチによる効果の検証が表 2 である．多重レベルディスパッチをしない場合，フラグメントのサイズが大きくなるにつれて GPU スレッドに待ち時間が発生することが想定され，16×16 以上の最大フラグメントサイズで実行速度が低下している．多重レベルディスパッチを使用した場合は待ち時間が減るため，フラグメントのサイズの大きさに合わせて実行速度が高速化される．

5.2 SSQ

VR 酔いに関する評価として，マスクなし，イーグルフライト同様の黒いマスク，ぼかしマスク，提案法のそれぞれについて，プレイ前との比較を SSQ アンケートを用いて行った．実施したアンケートについて，t 検定を行い評価した結果が図 7 である．95%の信頼区間に対

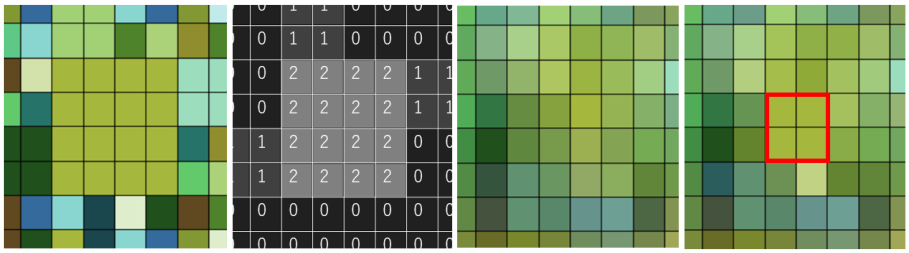

図 5 階層型線形補間の結果．左からカラーバッファ，階層の値，提案法による結果，通常の線形補間による結果．通常の線形補間では，階層の値が大きくなると，中央に他の色が混ざりこまない (赤枠内)．

フィルタなし	フィルタなし (CS)	黒マスク	ぼかしマスク	提案法	提案法 (期待値)
81.7 fps	17.0 fps	38.0 fps	79.4 fps	26.3 fps	126.2 fps

表 1 各手法における実行速度．右から，VR 酔い対策なし，VR 酔い対策がない場合を Compute Shader で実装，Eagle Flight と同様のマスク処理，CHIBA マスクと同様のぼかし処理，提案法，提案法 (Compute Shader 部分の改善が施された場合)．

して、マスクなしの結果はふらつき感 (D), 総合点 (TS) に有意差があり，眼精疲労 (O) には有意傾向が見られた ($\{D = 0.00004, TS = 0.002, O = 0.053\}$)．すなわち，本検証に用いたゲームは，VR 酔い対策が必要なアプリケーションであることが確認された．それぞれの手法に関して評価すると，黒いマスクでは，すべての項目で有意差はなかった．提案法では，ふらつき感 (D) のみ有意差 ($D = 0.016$)) があったが，他の項目 には有意差はなかった．ぼかしマスクでは，ふらつき感 (D) のみ有意差のある傾向 ($D = 0.010$) があったが，他の項目 には有意差は見られなかった．ゆえに，全ての手法に関して，ふらつき感に関する改善は必要であるが，VR 酔いの改善が行えていることが検証された．

5.3 品質アンケート

我々は 3 つのフィルタリングの方法についてプレイ前とプレイ後の結果を比較した (図 8)．マスク無しと黒いマスクの結果を比較したところ没入感に関する項目に優位傾向 ($p = 0.078$) が見られた．マスクが無しと Blurred マスクの結果を比較したところスピード感に関する項目に有意差 ($p = 0.037$) が見られたが他の項目に有意差はなかった．提案法では没入感と有意差 ($p = 0.023$) があり，画質についての項目に優位傾向 ($p = 0.074$) が見られた．以上の結果から提案法ではぼかしマスクと比較して画質に関する違いはないが没入感に対する効果は弱いことが判明した．

6. まとめ

VR での描画に関して，VR 酔いが起きないように状況に応じて詳細度を下げて実行速度を向上させると共に，フレームごとに詳細度を左右で入れ替えることによって，品質の低下を抑えるレンダリング手法を提案した．既存の手法と比較した結果，VR 酔い対策として提唱されているぼかしマスクよりも描画負荷が軽減されている．提案法を SSQ アンケートで評価した結果，VR 酔い対策としての効果があることが確認された．品質アンケートにおいてもマスク無しの結果と比較して一部を除いて有意差はなく，ゲームプレイにおける品質には問題がない．以上の結果から，提案手法を用いることで高速な描画処理を行いつつ VR 酔い対策として有用性があることが確認された．

本研究は，DCPS を拡張して実装を行ったが，Compute Shader で実装を行っているため，ハードウェア的に十分に効率化されているとは言い難く，将来の GPU での調査やアルゴリズムの改善が必要である．

参考文献

[1] LaViola, J. J. Jr, “A discussion of cybersickness in virtual environments,” ACM SIGCHI Bulletin. 32, pp.4–56, 2000.
[2] Guenter, Brian and Finch, Mark and Drucker, Steven and Tan, Desney and Snyder, John, “Foveated 3D Graphics,” ACM Trans. Graph., 31(6), pp.164:1–164:10, 2012.
[3] He, Yong and Gu, Yan and Fatahalian, Kayvon, “Extending the Graphics Pipeline with Adaptive, Multi-Rate Shading,” ACM Trans. Graph., 33(4), pp.142:1–142:12, 2014.
[4] Clarberg, Petrik and Toth, Robert and Hasselgren, Jon and Nilsson, Jim and Akenine-Möller, Tomas, “AMFS: Adaptive Multi-frequency Shading for Future Graphics Processors,” ACM Trans. Graph., 33(4), pp.141:1–141:12, 2014.
[5] Stengel, Michael and Grogorick, Steve and Eisemann, Martin and Magnor, Marcus, “Adaptive Image-Space Sampling for Gaze-Contingent Real-time Rendering,” Computer Graphics Forum, 35(4), pp.129139, 2016.
[6] Duchowski, Andrew T. and Çöltekin, Arzu, “Foveated Gaze-contingent Displays for Peripheral LOD Manage-

	2 × 2	4 × 4	8 × 8	16 × 16	32 × 32
多重レベルディスパッチなし	17.1 fps	20.0 fps	25.0 fps	19.0 fps	9.7 fps
多重レベルディスパッチ使用	17.0 fps	20.0 fps	27.0 fps	33.0 fps	37.0 fps

表 2 ポイントライト 100 個の環境における多重レベルディスパッチの利用の差異.

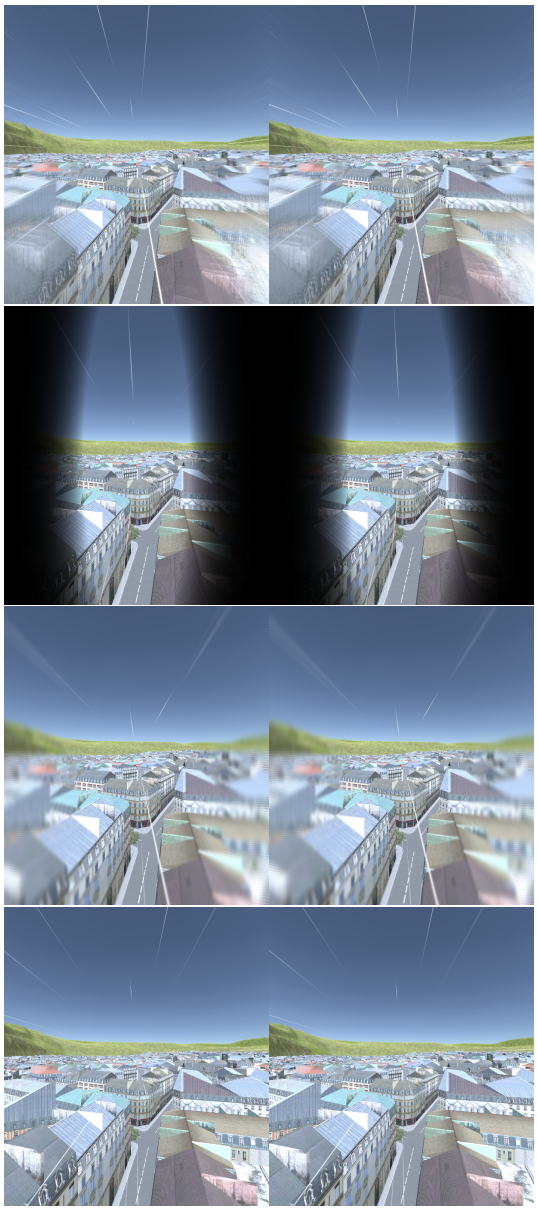

図 6 上から，提案法，黒いマスク，ぼかしマスク，通常の描画.

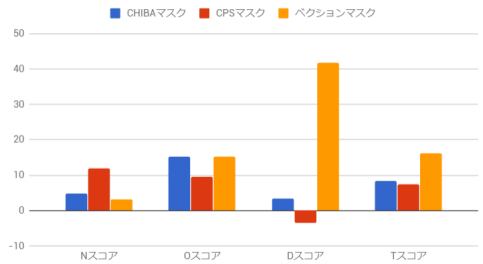

図 7 プレイ後の SSQ アンケートの平均と 95%信頼区間.

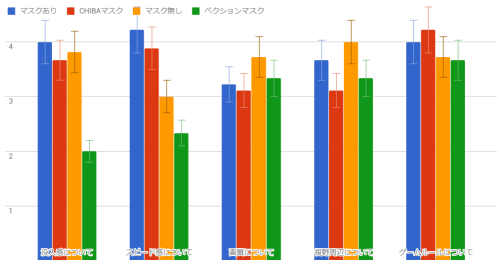

図 8 品質アンケートの平均と標準偏差.

ment, 3D Visualization, and Stereo Imaging," ACM Trans. Multimedia Comput. Commun. Appl., 3(4), pp.6:1–6:18, 2007.

[7] NVIDIA, "VRWorks-Lens Matched Shading," Retrieved from https://developer.nvidia.com/vrworks/graphics/lensmatchedshading, 2016.

[8] NVIDIA, "VRWorks-Multi-Res Shading," Retrieved from https://developer.nvidia.com/vrworks/graphics/multiresshading, 2016.

[9] James Answer, "Fast and Flexible: Technical Art and Rendering For The Unknown," GDC 2016 VRDC session, 2016.

[10] "Eagle Flight," Ubisoft, 2016.

[11] 千葉 瑞希, 中村 陽介, 渡辺 大地 and 三上 浩司, "ベクションを考慮したマスクの応用による VR 酔いの軽減と没入感の向上," 日本デジタルゲーム学会 年次大会 (2016).

[12] Vaidyanathan, K. and Salvi, M. and Toth, R. and Foley, T. and Akenine-Möller, T. and Nilsson, J. and Munkberg, J. and Hasselgren, J. and Sugihara, M. and Clarberg, P. and Janczak, T. and Lefohn, A., "Coarse Pixel Shading," Proceedings of High Performance Graphics, 10, pp.9–18, 2014.

[13] Rahul P. Sathe and Tomasz Janczak, "Deferred Coarse Pixel Shading," GPU Pro 7, CRC Press, pp.145–153, 2016.

[14] Patney, Anjul and Salvi, Marco and Kim, Joohwan and Kaplanyan, Anton and Wyman, Chris and Benty, Nir and Luebke, David and Lefohn, Aaron, "Towards Foveated Rendering for Gaze-tracked Virtual Reality," ACM Trans. Graph., 35(6), pp.179:1179:12, 2016.

[15] Jorge Jimenez and Jose I. Echevarria and Tiago Sousa and Diego Gutierrez, "SMAA: Enhanced morphological antialiasin," Computer Graphics Forum (Proc. EUROGRAPHICS 2012), 31(2), 2012.

[16] Brian Karis "High-quality temporal supersampling," In Advances in Real-Time Rendering in Games, SIGGRAPH Courses, 2014.

[17] Johansson G., "Studies on visual perception of locomotion," Perception, 6(4), pp.365–376, 1977.

[18] Robert S. Kennedy and Norman E. Lane and Kevin S. Berbaum and Michael G. Lilienthal, "Simulator Sickness Questionnaire: An Enhanced Method for Quantifying Simulator Sickness," The International Journal of Aviation Psychology, 3(3), pp.203-220, 1993.

[19] Tatsuya A, "Creation of immersive in entertainment using VR technology," master thesis, 2015.