令和 7 年度　　卒業研究

視覚効果による視線誘導が Redirected Walking に及ぼす影響に関する研究

電子制御工学科 5 年　矢吹 隼人

指導教員　　嶋田　英樹

目次

第 1 章　緒論

近年，ヘッドマウントディスプレイ（HMD）は価格の低下と対応コンテンツの増加に伴い，幅広い用途で利用されるようになってきた．教育，訓練，医療，エンターテインメントなど多様な分野での応用が進み，没入型インタラクションの基盤技術として VR（仮想現実）が社会実装に近づいている．HMD を用いた VR の利点の一つは，利用者が仮想空間内を身体的に移動できる点である．実際に歩いて移動することで，視覚・前庭感覚・固有受容感覚の一致が高まり，没入感と行動的自然さが向上する．

しかし現状では，ユーザが自由に移動できる物理空間（トラッキング空間）は限定される場合が多く，室内環境では数平方メートル程度の空間しか確保できないことが一般的である．この制約は，仮想空間における情景のスケール感や探索自由度を制限し，歩行移動に基づく自然な体験の提供を阻害する要因となる．多くのアプリケーションはこの問題に対処するためにテレポートやスティック操作といった代替的な移動手法を採用しているが，これらは身体運動と視覚情報の不一致を生み，没入感低下や酔いの原因となることが知られている．

そこで，利用者が実際に歩く「身体的な移動感」を保ちながら狭い物理空間を有効利用するためのソフトウェア的手法として，Redirected Walking（RDW）が提案されている [1]．本研究は，RDW の基本原理を踏まえつつ，視覚的な副刺激（視覚効果）を組み合わせることで RDW の有効範囲と実用性を拡張し，違和感や酔いの発生を抑制することを目的とする．

本論文の構成は以下のとおりである．第 2 章では，Redirected Walking（RDW）の必要性と基本原理について述べ，直進移動および回転移動に対する各種スケーリング手法を整理するとともに，RDW の課題点とその改善に関する先行研究を概略を述べる．第 3 章では，視覚効果の有無による RDW の比較実験について述べ，実験環境，実装方法，実験条件および結果を示し，視覚効果による RDW の改善効果を検証する．最後に，第 4 章では本研究の結論を述べ，今後の課題について言及する．

第 2 章　Redirected Walking（RDW）について

Redirected Walking（RDW）は，仮想空間内ではユーザが直進していると知覚する一方で，実空間では歩行方向を徐々に変化させる誘導技術である．本手法により，限られた実空間内においても，広大な仮想空間を歩行しているかのような体験を提供することが可能となる．以後での内容を示す．

2.1　RDW の必要性

HMD は利用者に強い没入感を与えるデバイスであり，視点や手の動きが仮想空間に直結することで自然な操作が可能になる．しかし，歩行を伴う自然な移動（ナチュラル移動）を実現するには十分な物理空間が必要であり，多くの利用環境ではこれが確保できない．したがって多くのアプリケーションではテレポートやゲームパッドによる移動に頼らざるを得ない．歩行を可能にするために大型のモーションプラットフォームや電動トレッドミルを導入する選択肢もあるが，これには高いコストと設置スペースが必要であり，普及の障壁となる．図 2.1 に示すような商用の歩行プラットフォーム（例：KAT Walk C2+）は技術的には有効であるが，コストや運用性の観点から一般普及には限界がある．

図 2.1　KAT Walk C2+（カットウォークシーツープラス）

一方で，RDW は追加のハードウェアを必要とせず，ソフトウェア側の操作のみで自然な歩行体験を拡張できる点が利点である．ユーザの HMD の表示を微量ずつ操作することで，ユーザの進行方向を微妙にずらし，身体が実際に誘導されて歩いていることをユーザに認識させずに，広い仮想領域を探索させる．これはソフトウェア中心の解法として効率が高く，既存の HMD 環境に容易に適用できる．

2.2 RDW の原理

RDW では，ユーザの HMD に操作を加えることでユーザの移動を誘導する．ユーザの向きや動きから現実空間で移動する方向を予測し，適切な誘導を行うことで広い仮想空間を移動していると錯覚させる．RDW で加える典型的な操作は以下のとおりである[2].

2.2.1 直進移動量のスケーリング

図 2.2 にイメージ図を示す. 本手法はユーザが実際に進んだ距離に対して表示上の移動量を増減させるものである．例えばユーザが 1 メートル進んだ際に仮想空間上では 1.5 メートル進むように表示することで，実空間での移動量を節約する．

2.2.2 回転量のスケーリング

図 2.3 にイメージ図を示す. 本手法はユーザの頭部回転に対して仮想カメラの回転量をわずかに増減させるものである. 実際の回転量に対して仮想世界の回転量を微小に拡大・縮小することで, ユーザは自身が向きを変えたと認識していない場合でも, 身体の向きが実際には微妙に変化する. 例えば右方向への誘導を行う際には, 右向きの頭部回転に対して回転利得を大きくし, 左向きの回転に対して減衰させることで, 結果的にユーザの向きが右側へ傾くように制御する.

2.2.3 直進移動への曲率付加

図 2.4 にイメージ図を示す．本手法はユーザが直進している間に，物理空間上の移動軌跡に微小な曲率（カーブ）を付加するものである．これによりユーザは仮想空間では直進していると認識し続けながら，実際には円弧状に歩行している状態を作り出すことができる．速度に応じて曲率の半径を調整することで，不自然さを抑えつつ効果的な誘導が可能となる．

2.2.4 カーブ移動時の曲率スケーリング

図 2.5 にイメージ図を示す．本手法はユーザの歩行経路の曲率を増減し，ユーザの曲線移動の曲がり具合を調整し進む方向を誘導できる．ユーザがすでに曲線を歩いている状況において, その曲率の大きさを増減することで進行方向の誘導効果を高める手法である. 具体的には ユーザが実際に曲がっている量と仮想空間上で感じる曲がり量に差異を付けることで, 方向転換の程度を操作しやすくする. この操作により，曲がる感覚のピークやタイミングを調整しやすくなり，より自然で制御された誘導が可能となる．

図 2.2 直進移動量のスケーリング

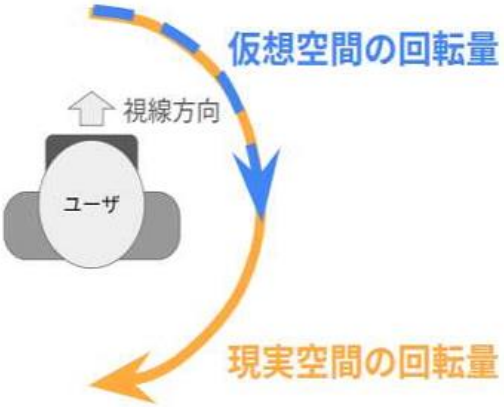

図 2.3 回転量のスケーリング

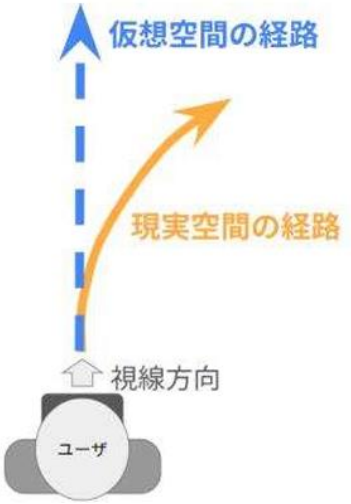

図 2.4 直進移動への曲率付加

図 2.5 カーブ移動時の曲率スケーリング

2.3 RDW の課題点

RDW はユーザの進行方向を違和感のない程度に微小量の操作を加えることで実現している. つまり, 操作量を拡大するとより大きな物理領域を仮想的に再現できる. しかし RDW の操作量を大きくするほど，HMD の視界上と，実際の移動が異なるため視界情報と前庭（内耳）情報との齟齬が大きくなる．視覚情報と前庭系（内耳の平衡感覚）による角速度・加速度感覚との不一致は, 違和感やシミュレーター病(いわゆる VR 酔い)を誘発しやすい. したがって，操作量は知覚閾値の範囲内に抑える必要があり，これが RDW の適用範囲を制限している．先行研究では，RDW を適応した状態の仮想空間を違和感なく直進し続ける

ために実空間上に半径 22m 以上の円弧上の経路を用意する必要があるとされており[3], 非常に広い空間を用意する必要がある.

2.4 RDW の補助の先行研究

RDW の問題点を改善するために, 先行研究では RDW の違和感を軽減するための方法が検討されている.

例えば, 視覚情報のみで誘導を行う従来の RDW に対し, 視覚と触覚の両方を用いる拡張も試みられている. この研究ではユーザが壁面を触り触覚提示を与えることで, 並進方向の誘導を視覚だけに頼らず実現する手法である. この研究では, 円形の壁に触った場合の有効性を実験的に検証されている. 実験結果から, 触覚提示を併用することで視覚のみの RDW と比べて空間知覚操作の効果が向上し, 誘導効果が強化される可能性が示された[4].

また, 手押し車の角度制御で触覚フィードバックを得ながら RDW を実装し, より自然な誘導を狙う研究なども行われている[5].

このような触覚を用いた RDW の補助は誘導効果が強化される可能性がある反面, 外部に物体や機器を置かなければならず, 容易さや汎用性を損なっている. ソフトウェアの改良により RDW を補助できるならそれが最も容易である.

第 3 章　視覚効果の有無による RDW の比較実験

本章では，視覚効果を用いた視線誘導が Redirected Walking（RDW）の主観的評価に与える影響について検証する．まず，回転ゲインを用いた RDW の実装方法と，視線誘導を目的とした各種視覚効果の設計について述べる．次に，これらを実装した仮想空間での歩行実験を実施し，歩行経路，視線方向の変化，および主観評価値の観点から，視覚効果が RDW の違和感低減に影響する可能性を検討する．

3.1　視点の移動量と RDW の関連性

本研究では，視覚効果による視線誘導が RDW の効果を補助し，利用者が感じる違和感を低減できるかを検証する． 今回は，回転ゲイン（rotation gain）を主要な操作対象とした．回転ゲインはユーザの頭部回転量に応じて視界をわずかに加速度的に回転させ，結果としてユーザの向き変化量を増幅または減衰させる．視線誘導（視覚効果）は，利用者の視線分布を操作して，意図する方向への注視を促し，結果として頭部回転の発生頻度や振幅に影響を与えることが期待される．このため，視点移動が増加した場合，仮想的に与える回転ゲインの効果は視覚的には自然に感じられやすく，前庭系情報との不一致が相対的に小さくなると仮定した．

3.1.1　適応する視覚効果

視覚誘導の基本的な考え方は，人間の視覚システムが周辺視野の低次情報に敏感であるという性質を利用し，視線方向を操作するものである．実際，視覚的周辺情報の微細な変化を用いることでユーザを無意識の内に誘導することが可能である[6].

さらに，気づかれない視線誘導のために解像度の変化や視覚刺激の強度に関する研究が行われており，知覚閾値以下の視覚変化でも視線誘導が成立する可能性が示されている．この研究は，視覚誘導効果と知覚の関係を実験的に明らかにしている[7].

これらの先行研究を踏まえ，本研究では視線誘導を目的とした複数の視覚効果条件を実験に採用した．具体的には，視覚効果なしを基準条件とし，次の三つの視覚的演出を誘導刺激として用いた：(1) 点滅，(2) 片側ブラー（画面の一部へのぼかし），(3) 片側彩度上昇（色の鮮やかさの強調）である．これらの効果は，ユーザの注意を視野内の特定方向へ向けさせることを意図して設計した．点滅は強い視覚刺激として注意喚起効果が高いと想定される一方，ブラーや彩度変化は視覚的顕著性を変化させることで比較的穏やかに注視誘導を行う効果が期待される．

3.1.2　仮想空間

直進歩行を行うための仮想空間として図 3.1 に示す，RDW を適応可能な仮想空間を unity で作成した．作成した仮想空間は，適切に直進歩行できるように，細い一本道とした．没入

感の向上と視覚効果を適応させやすくするため，背景は現実の街を再現したものにした．

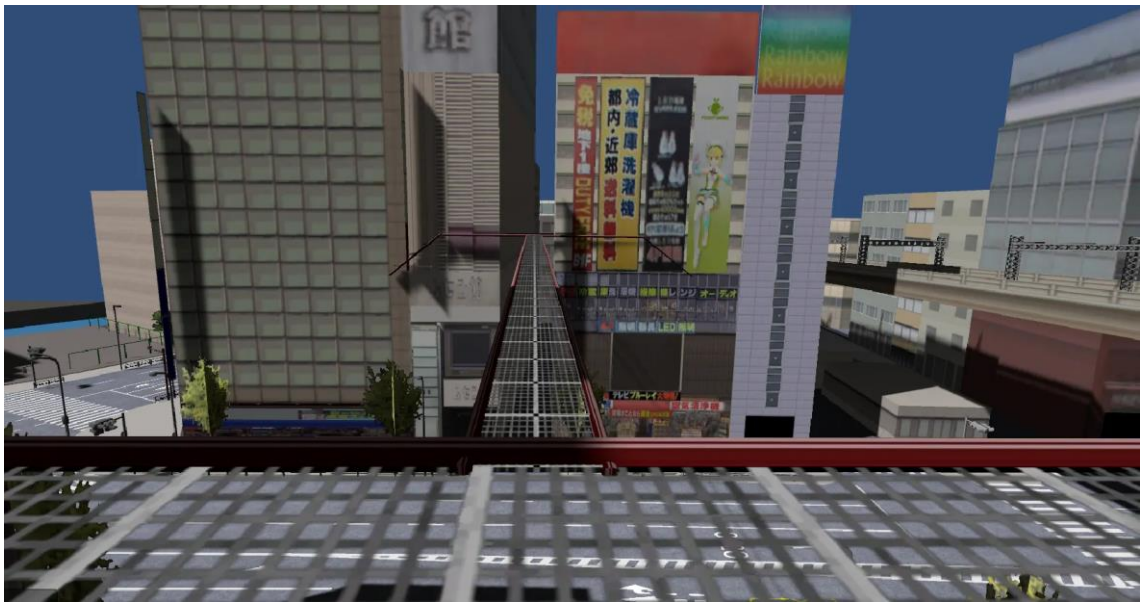

図 3.1 Unity で製作した直進歩行用の仮想空間

3.1.3　RDW の実装

RDW の実装は Orientation-to-Curvature という，ユーザの身体の向きを，歩行速度に応じて歩行経路に一定半径の曲率を与える方式で実装した．具体的なプログラムを以下に示す．このプログラムは，HMD の頭部の座標から水平成分の速度を算出し，速度が変化しても曲率半径が一定である角速度をプレイヤーに与えることで RDW を適応している．

```
    // 1）速度を近似（m/s）
    Vector3 delta = new Vector3(targetTransform.position.x – prevPos.x , 0f ,
targetTransform.position.z – prevPos.z );
    float speed = delta.magnitude / Mathf.Max(Time.deltaTime, 1e-6f);

    // 速度がほぼ 0 のときは補正しない
    if (speed < 1e-3f)
    {
        prevPos = targetTransform.position;
        return;
    }

    // 2）必要な角速度（rad/s）= v / r
    float angVelRad = speed / Mathf.Max(radius, 1e-6f); // rad/s
    float angDegPerSec = angVelRad * Mathf.Rad2Deg;
```

```
// 3）右回り（clockwise）なら負の角度（Unity の Y 回転は左回りが正）
    float sign = clockwise ? -1f : 1f;
    float angleThisFrame = sign * angDegPerSec * Time.deltaTime;

    // 5）プレイヤーの回転に適用（Y 軸回転）
    rig_target.transform.RotateAround(
        targetTransform.position,    // 回転の中心（A の座標）
        Vector3.up,                  // 回転軸（Y 軸回転）
        angleThisFrame// 毎フレームの回転量
    );
```

3.1.4　視覚効果

提示した視覚効果は次の通りである．

点　滅　：画面端に短時間で強い明暗コントラストの点滅を複数回呈示し，反射的に視線を注目させる．図 3.2 に示す仮想空間では画面右端に赤色の点刺激を短時間表示する操作を複数回繰り返し，画面右端への視線誘導を試みた．以下にこの視覚効果を実装したプログラムを示す．本プログラムでは約 80ms 間隔で赤色の点刺激を 2ms 間表示する操作を 4 回繰り返している．

```
private bool eA = false;
 async public void EffectA()
 {
      //フラグ管理
      if (eA) return;
      eA = true;

      //視覚効果の発生開始を記録
      GazeDataRecorder.writeRed_point(1, 1);

      //点滅
      for (int i = 0; i < 4; i++)
      {
          effectA.SetActive(true);
          await Task.Delay(2);
```

```
            effectA.SetActive(false);

            await Task.Delay(80);
        }

        //視覚効果の終了を記録
        GazeDataRecorder.writeRed_point(1, 0);
        eA = false;
    }
```

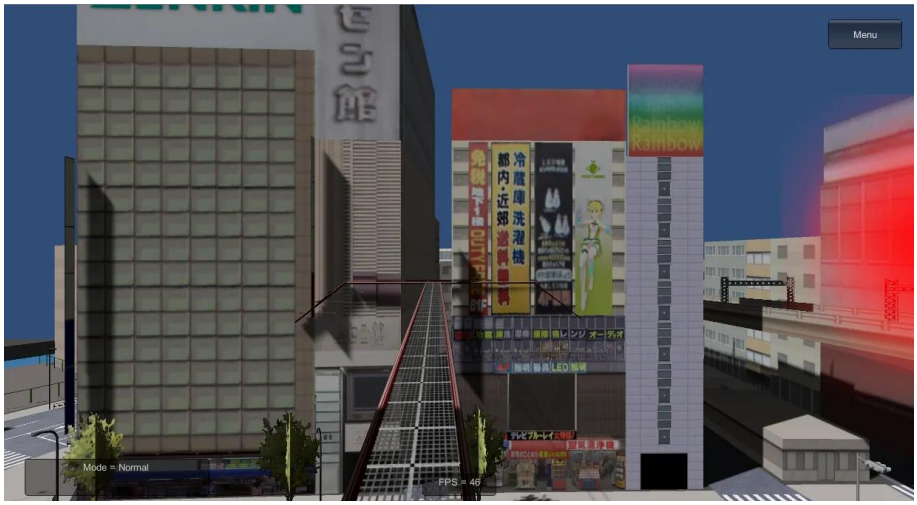

図 3.2 画面右端での点滅

ブ ラ ー ：画面の片側に局所的なぼかしを発生させ，視線を反対側に向ける操作を行う．図 3.3 に示す仮想空間では画面左側に一時的にブラー処理を適応し，画面右側への視線誘導を試みた．以下にこの視覚効果を実装したプログラムを示す．本プログラムではブラー強度を約 700 ms かけて徐々に増加させ，その後約 1200 ms 維持した後，約 1050 ms かけて段階的に減少させる制御を行った．これにより，視覚的違和感を抑えつつブラー処理を実現した．

```
private bool eB = false;
async public void EffectB()
{
    //フラグ管理
    if (eB) return;
    eB = true;

    float b;
```

```
    Material mat;

     effectB.SetActive(false);
     effectB.SetActive(true);
     mat = effectB.GetComponent<Renderer>().material;
     b = mat.GetFloat("_BlurAmount");

     for (int i = 7; i > 0; i--)
     {

          //ブラーの強さを初期値に向けて変化させていく
          mat.SetFloat("_BlurAmount", b / ((((float)i * (float)i) + 9 * (float)i)
/ 10f));

          //視覚効果の発生開始を記録
          GazeDataRecorder.writeBlur(b, b / ((((float)i * (float)i) + 9 *
(float)i) / 10f));

          await Task.Delay(100);
     }

     await Task.Delay(1200);
     for (int i = 1; i <= 7; i++)
     {
          //ブラーの強さを初期値に向けて変化させていく
          mat.SetFloat("_BlurAmount", b / ((((float)i * (float)i) + 9 * (float)i)
/ 10f));



          //視覚効果の終了を記録
          GazeDataRecorder.writeBlur(b, b / ((((float)i * (float)i) + 9 *
(float)i) / 10f));

          await Task.Delay(150);
     }
```

```
    eB = false;

     mat.SetFloat("_BlurAmount", b);
     effectB.SetActive(false);
     GazeDataRecorder.writeBlur(b, 0);
}
```

図 3.3 画面左側へのブラー

彩度上昇：画面の片側の彩度（色の鮮やかさ）を上げ，注目を誘導する．図 3.4 に示す仮想空間では，画面右端領域の彩度を一時的に上昇させる操作を複数回表示し，画面右端への視線誘導を試みた．以下にこの視覚効果を実装したプログラムを示す．本プログラムでは彩度を約 700 ms かけて徐々に増加させ，その後約 1200 ms 維持した後，約 1050ms かけて段階的に減少させる制御を行った．これにより，視覚的違和感を抑えつつ彩度上昇処理を実現した．

```
    private bool eC = false;
    async public void EffectC()
    {
        //フラグ管理
        if (eC) return;
        eC = true;

        float b;
        Material mat;
```

```
            effectC.SetActive(false);
            effectC.SetActive(true);

            mat = effectC.GetComponent<Renderer>().material;
            b = mat.GetFloat("_Saturation");

            for (int i = 7; i > 0; i--)
            {
                //彩度を初期値に向けて変化させていく
                mat.SetFloat("_Saturation", b / ((((float)i * (float)i) + 9 *
(float)i) / 10f));

                //視覚効果の発生開始を記録
                GazeDataRecorder.writeSaturation(b, b / ((((float)i * (float)i) +
9 * (float)i) / 10f));

                await Task.Delay(100);
            }

            await Task.Delay(1200);
            for (int i = 1; i <= 7; i++)
            {
                //彩度を初期値に向けて変化させていく
                mat.SetFloat("_Saturation", b / ((((float)i * (float)i) + 9 *
(float)i) / 10f));

                GazeDataRecorder.writeSaturation(b, b / ((((float)i * (float)i) +
9 * (float)i) / 10f));

                await Task.Delay(150);
            }

            eC = false;

            mat.SetFloat("_Saturation", b);
            effectC.SetActive(false);
```

```
//視覚効果の終了を記録
            GazeDataRecorder.writeSaturation(b, 0);
    }
```

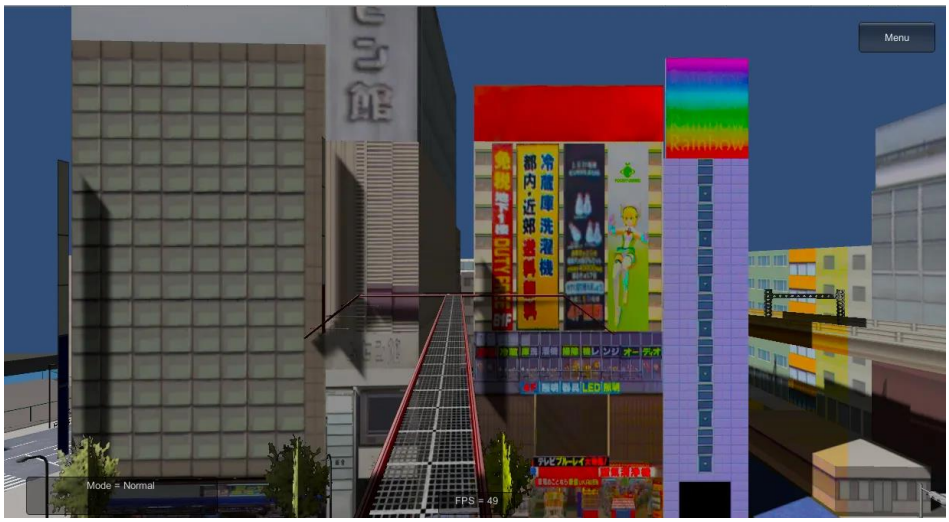

図 3.4 画面右側の彩度上昇

3.2 視覚効果による改善の検証

3.2.1 実験の概要

図 3.5 に実験手順の模式図を示す．被験者は HMD を用い，仮想空間内を一定速度で直進歩行する間に，視覚効果が数回提示される．この手順を視覚効果条件 7 種類（視覚効果なし，点滅，ブラー，彩度上昇，点滅＋ブラー，点滅＋彩度上昇，ブラー＋彩度上昇）について行い，歩行終了後，被験者は自身がどの程度曲がったと感じたかを，視覚効果なしの条件を 5 とした場合の 0 から 10 の主観評価値として回答してもらった．このとき，0 と回答した場合は実空間でも直進したと感じたことを示す．また，本実験を行った 11 人の被験者の実験中の頭部位置および視線方向をリアルタイムで取得した．

この実験結果から，視覚効果なし条件に比べ視覚効果あり条件の主観評価値が低い傾向が確認された場合，視覚効果の存在が RDW の曲率変更操作の違和感を軽減する可能性が示される．また本実験で取得した歩行実験中の頭部位置および視線方向から，実空間の歩行経路が円弧の軌跡を辿ることや，視覚効果が視線誘導できているかを確認し，仮定の論理を補強するデータとする．

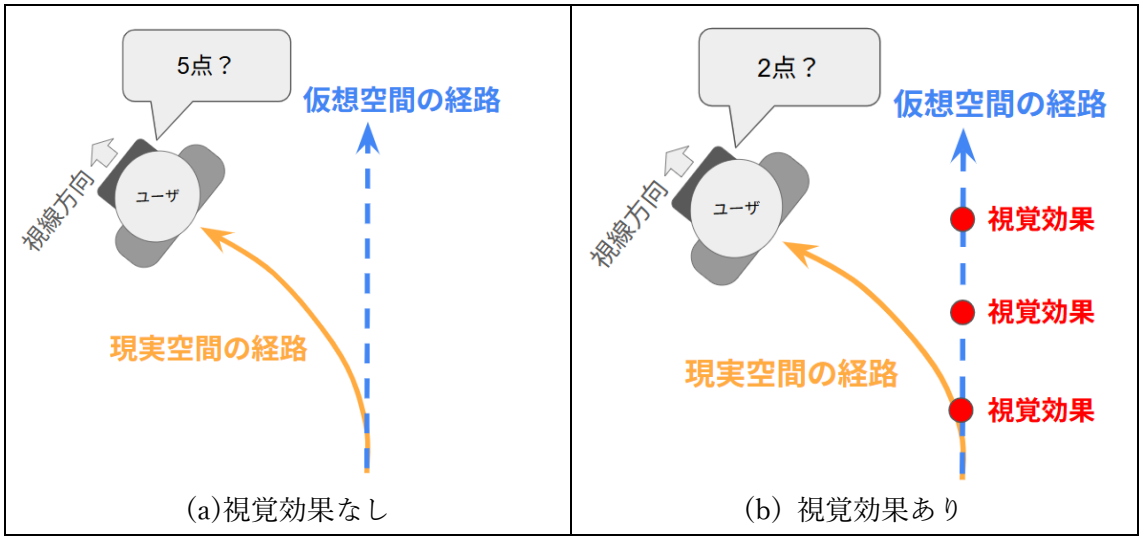

図 3.5 実験手順の模式図

3.2.2 実験条件

第 3.2.1 項にて示した実験を行うため，以下のような制御処理，機器，環境を用意した．

・制御処理

速度に比例した曲率の RDW を実装した．どの速度で歩行しても，一定の半径で曲がるようにした．時間ごとの頭部の位置・回転データと，視覚効果の発生時刻を記録し，表に出力できるようにした．

・機器・環境

Meta Quest 3 を使用し，屋内の平坦な空間で実験を実施した．

3.2.3 実験の手順

①被験者は HMD(Meta Quest 3)を装着する．
②仮想空間上の出発点へ視点位置をリセットする．
③被験者は仮想空間内をゆっくり直進歩行する．
④特定距離間隔ごとに視覚効果を発生させる．
⑤ある程度歩いた後，停止し，どのくらい曲がったかの度合いを 0~10 点で回答する．
⑥視覚効果を変更し，②に戻る

ここで，⑥の視覚効果は，（視覚効果なし，点滅，ブラー，彩度上昇，点滅＋ブラー，点滅＋彩度上昇，ブラー＋彩度上昇）について実施した．

3.2.4　実験結果

実験手順に従って取得したデータを基に，以下の 3 点について検証する．

1. 歩行経路が実際に曲がっているか
2. 視覚効果によって視線誘導が生じているか
3. 視線誘導の成否と主観評価値との関係

本実験では，各時刻における被験者の頭部位置（座標），視線方向（角度），および視覚効果条件を記録した．加えて，各条件終了後に主観のアンケート調査を行った．

以下では，各実験結果に対して考察する．

考察 1. 歩行経路は曲がっているか

図 3.6 に左向きに湾曲した歩行の軌跡の一例を示す．この図は被験者の歩行経路を上方視点から投影した軌跡であり，仮想空間内の歩行軌跡は直線に近しいのに対し，実空間の歩行軌跡は左に大きく湾曲している．これは RDW が適応され，直線移動に曲率が付与された結果である．

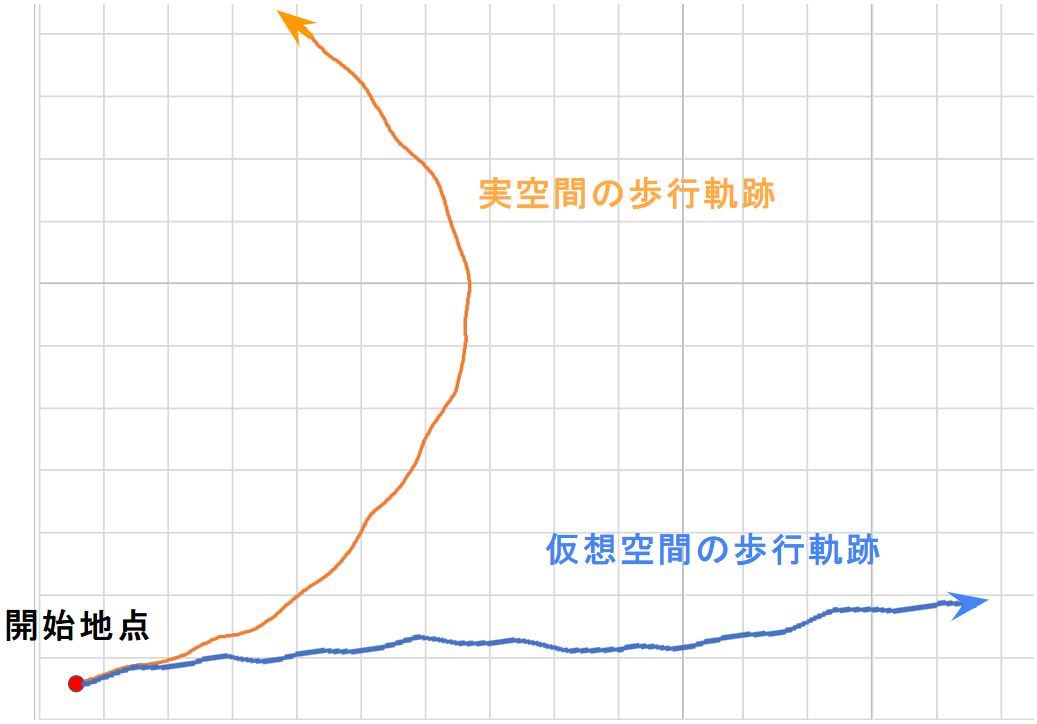

図 3.6 左向きに湾曲した歩行軌跡の一例

図 3.7 と図 3.8 に示す視線方向の時系列変化の例のように，視線方向の値が徐々に変化する傾向があった．このことから，RDW による操作により，被験者の視点方向が段階的に変化していた可能性が示唆された．図 3.7 の後半部分に瞬間的に 300° 以上視点方向が変化している箇所があるが，これは 0° を下回った視点方向の値が 360° に移行しているためであ

り，実際に大きな視点方向の変化があったわけではない．図 3.7 で視覚効果が発生した直後に視点方向が山なりに盛り上がっている部分は，提示された視覚効果に対する反応として視線誘導が生じた結果であると考えられる．一方で，図 3.8 では視覚効果が発生した直後に視点方向へ特徴的な変化がなく，視覚効果による視線誘導ができていないと考えられる．

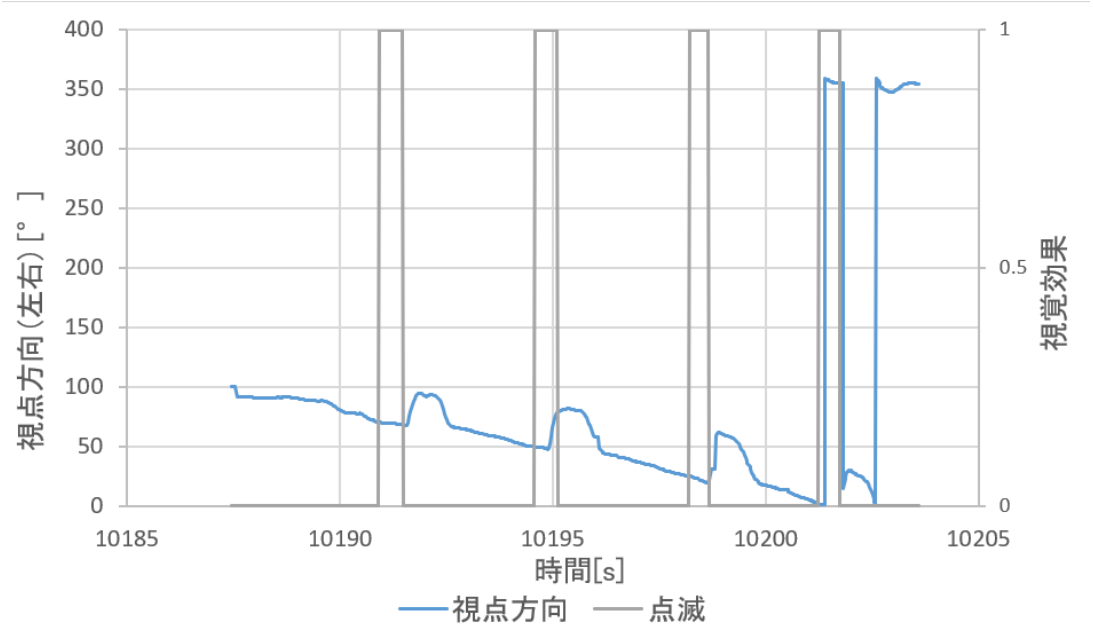

図 3.7 視線方向(左右)と視覚効果(点滅)の時系列変化の一例

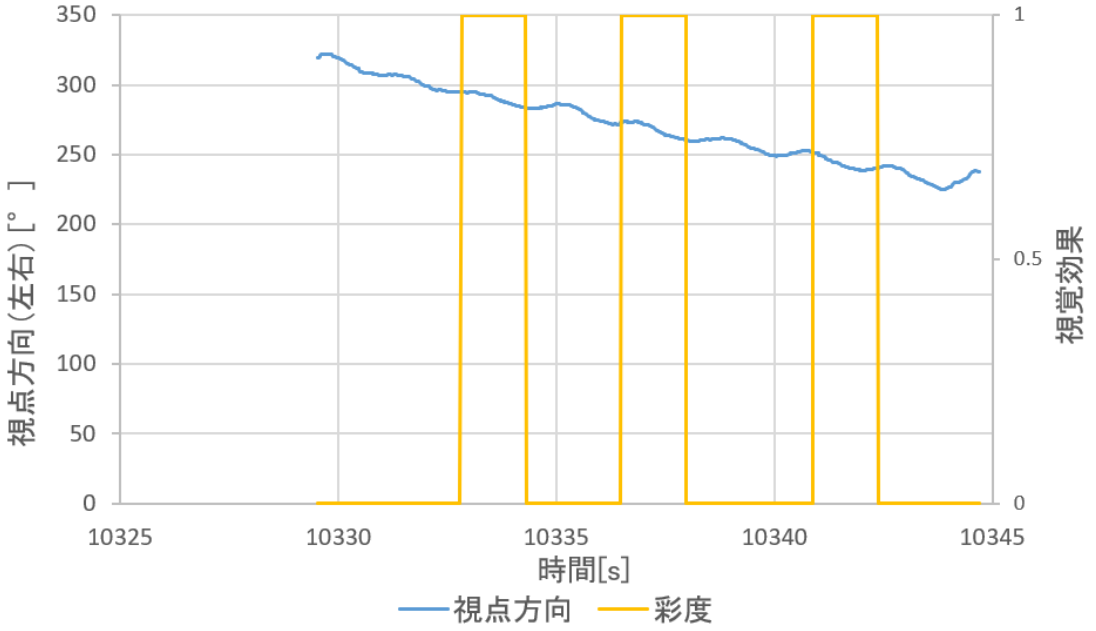

図 3.8 視線方向(左右)と視覚効果(彩度上昇)の時系列変化の一例

考察 2. 視覚効果によって視線誘導は生じているか

表 1 は，各実験条件において視覚効果提示後に視線方向の波形に特徴的な変化が確認されたかを示すものであり，1 は変化が観察された試行，0 は変化が観察されなかった試行を表す．全体として約 59%の試行で視線方向が実験者の意図した方向へ偏向する傾向が観測された．視覚効果別に見ると，「点滅」「点滅＋ブラー」「点滅＋彩度上昇」「ブラー＋彩度上昇」条件では約 63〜82%の試行において視線誘導反応が確認され，一方で単独の「ブラー」条件は約 18%，「彩度上昇」条件は約 36%にとどまり，相対的に誘導反応が弱い傾向が示された．

この結果の解釈としては，まず本研究で用いた視覚効果の多くが「無意識下での視線誘導」を目標としており，色の濃さやコントラストなどに代表される視覚的な目立ちやすさ（以下，視覚効果の強度）を過度に高めない設定としたため，単独条件では知覚閾値を下回り誘導が十分に発現しなかった可能性が考えられる．逆に，複合条件（ブラーと彩度上昇の同時提示）で誘導反応が比較的高かったのは，二つの刺激が重複することで視覚効果の強度が高まり，結果として視線誘導の発生確率が上がったためと推測される．したがって，単独のブラーや彩度上昇条件についても，刺激パラメータ（ぼかしの強さや半径，彩度の上昇量など）を段階的に増加させれば，誘導反応を強化できる可能性が示唆される．

また，被験者間で視覚効果に対する反応の程度に差が認められた．この要因として，視力特性の違いに加え，実験中の HMD の装着位置のずれや，視覚刺激に対する知覚閾値の個人差などが影響している可能性が考えられる．

表 1 各視覚効果による視線の右方向変化の有無

	点滅	ブラー	彩度	点滅+ブラー	点滅+彩度	ブラー+彩度	
被験者A	1	0	0	0	0	1	
被験者B	1	0	0	0	0	0	
被験者C	0	0	0	1	1	1	
被験者D	1	0	0	1	1	0	
被験者E	1	0	1	1	1	1	
被験者F	1	0	0	1	1	1	
被験者G	1	0	1	1	1	1	
被験者H	1	1	1	1	1	0	
被験者I	1	0	0	1	1	0	
被験者J	0	0	0	0	1	1	
被験者K	1	1	1	1	1	1	全体
割合	0.8182	0.1818	0.36364	0.72727273	0.818182	0.63636364	0.590909

考察 3. 視線誘導の成功度と主観評価値の関係

視覚効果なし条件と各視覚効果条件における主観評価値の平均および中央値を表 2 に示す. 視線誘導が成功している条件は主観評価値が低下するという仮説を検証するため, 各被験者の評価値について視覚効果なし条件との対応のある t 検定を行った.

その結果, 全ての視覚効果条件において $p > 0.05$ となり, 視覚効果の有無による主観評価値の差は統計的に有意ではなかった. すなわち, 本実験においては, 視線誘導が生じていると考えられる条件であっても, 主観評価値が有意に低下するとは言えなかった.

表 2 各視覚効果条件の主観評価値

	なし	点滅	ブラー	彩度	点滅+ブラー	点滅+彩度	ブラー+彩度
被験者A	5	3	4	2	2	1	0
被験者B	5	0	0	0	4	0	2
被験者C	5	7	5	3	1	6	3
被験者D	5	1	1	6	7	4	6
被験者E	5	8	6	8	3	7	8
被験者F	5	6	7	6	8	8	7
被験者G	5	7	8	5	4	4	3
被験者H	5	7	6	4	4	3	2
被験者I	5	4	8	7	5	8	7
被験者J	5	7	7	9	8	7	8
被験者K	5	2	7	2	3	3	2
平均	5	4.7273	5.36364	4.72727	4.45454545	4.636364	4.36363636
中央値	5	6	6	5	4	4	3
t検定(両側)		0.7527	0.65913	0.74968	0.44838372	0.667566	0.471105

以上の結果から, 本研究で用いた視覚効果は一部の条件において被験者の視線を意図した方向へ偏向させうることが示唆されたが, その誘導効果は条件や被験者間で安定しておらず, 主観評価値に統計的に有意な改善は認められなかった. 具体的には, RDW 自体は実空間上で歩行経路に期待どおりの曲率を付与して機能していたものの, 視覚効果の多くを比較的低強度に設定したこと, 視覚刺激に対する個人差（視力・知覚閾値）が, 視線誘導の発生および主観評価への反映を妨げた可能性が高い. このことから, 本研究で用いた視覚効果による視線誘導は, RDW の動作そのものには影響を与える可能性がある一方で, 利用者が感じる違和感の低減に直接的に寄与するまでには至らなかったと考えられる.

4章　結論

本研究では，Redirected Walking（RDW）環境下において視覚効果を用いた視線誘導が歩行経路および主観評価に与える影響を検証した．実験では，被験者の頭部位置，視線方向，および視覚効果条件を時系列で取得し，歩行経路の曲率，視線誘導の成否，および主観評価値との関係について分析を行った．

まず，歩行経路の分析結果から，RDW による操作により被験者の歩行経路が意図した方向へ緩やかに湾曲していることが確認された．また，視線方向は歩行中に急激な変化を示すことなく連続的に変化しており，被験者が不自然な操作を意識することなく歩行していたと考えられる．これらの結果は，本実験環境において RDW が安定して機能していることを示している．

次に，視覚効果による視線誘導については，視線が意図した方向へ偏向する傾向が観測された．一方で，片側ブラーおよび片側彩度上昇を用いた条件では，他の視覚効果と比較して誘導反応が弱い傾向が見られた．無意識下での視線誘導を狙った視覚効果であっても，効果の種類によって誘導の成否に差が生じる可能性が示唆された．

視線誘導の成功度と主観評価値との関係について検証した結果，視覚効果なし条件と各視覚効果条件との間に有意な差は認められなかった．すなわち，本実験条件においては，視線誘導が生じていると考えられる場合であっても，主観評価値が有意に低下するとは言えなかった．

以上より，本研究では，RDW 環境下において視覚効果が視線誘導に影響を与える可能性は示されたものの，その効果が主観評価の改善に直結するとは結論づけられなかった．本実験では視覚効果による視線誘導が必ずしも安定して生じなかった．その結果，視覚効果を付与した条件においても，RDW の主観評価値が統計的に有意に低下するとは結論づけられなかった．

今後は，視覚効果の強度や提示位置などの条件を系統的に変化させた検証を行うとともに，アンケート調査を通じて被験者の視力特性や各視覚効果条件に対する主観的印象といった具体的なデータを収集する必要がある．これらの情報を踏まえて分析を行うことで，本研究において視覚効果を付与した条件でも RDW の主観評価値に有意な低下が見られなかった要因を明確化することで，今後の視覚効果設計および RDW 環境の最適化に向けた指針を得ることが可能になると考えられる．

謝辞

本研究を進めるにあたり，指導教員である嶋田先生には，実験計画，論文作成に至るまで，終始丁寧なご指導を賜りました．多忙な中にもかかわらず，親身にご助言いただきましたことに，心より御礼申し上げます．

参考文献

[1] Sharif Razzaque, Zachariah Kohn, and Mary C. Whitton, “Redirected walking,” Proceedings of Eurographics 2001 – Short Presentations, pp. 105-106, Sept. 2001.
[2] Frank Steinicke, Gerd Bruder, Luv Kohli, Jason Jerald, and Klaus H. Hinrichs, “Taxonomy and implementation of redirection techniques for ubiquitous passive haptic feedback,” in 2008 International Conference on Cyberworlds, pp. 217-223, Apr. 2008.
[3] Frank Steinicke, Gerd Bruder, Jason Jerald, Harald Frenz, and Markus Lappe, “Estimation of detection thresholds for redirected walking techniques,” IEEE Transactions on Visualization and Computer Graphics, vol. 16, no. 1, pp. 17-27, Jan. 2010.
[4] 松本啓吾，鳴海拓志，伴祐樹，谷川智洋，廣瀬通孝，“視触覚間相互作用を用いた曲率操作型リダイレクテッドウォーキング”, 日本バーチャルリアリティ学会論文誌, vol. 23, no. 3, pp. 129-138, 2018.
[5] 寺尾颯人，井尻敬，“手押し車による触覚フィードバックを利用した Redirected Walking”，芸術科学会芸術科学フォーラム論文録，2022.
[6] Reynold Bailey, Ann McNamara, Nisha Sudarsanam, Cindy Grimm，“Subtle gaze direction”，ACM Transactions on Graphics，vol. 28, no. 4, 2009.
[7] 畑元，小池英樹，佐藤洋一，“解像度制御を用いた視線誘導”，インタラクション 2014 論文集，pp. 57-64，2014.

付録

O2C.cs

```
using System.Collections; using System.Collections.Generic; using UnityEngine;
public class O2C : MonoBehaviour { [Tooltip("補正をかけたい対象の Transform (プレイヤー)")] public Transform targetTransform;
[Tooltip("補正をかけたい対象の Transform (プレイヤー)")]
public GameObject rig_target;

[Tooltip("曲率半径（m）．小さくするほど急に曲がる")]
public float radius = 0.6f;

[Tooltip("右回り(true) / 左回り(false)")]
public bool clockwise = true;

public bool isO2C =false;

// 内部：前フレームの座標を使って実速度を近似する（参照なしでスピード推定）
private Vector3 prevPos;

void Start()
{
    if (targetTransform == null)
        targetTransform = this.transform;
    prevPos = targetTransform.position;
}

// Update is called once per frame
void Update()
{
    if (isO2C == false) return;
    // 2）速度を近似（m/s）
    Vector3 delta = new Vector3(targetTransform.position.x – prevPos.x,0,
targetTransform.position.z – prevPos.z);
    float speed = delta.magnitude / Mathf.Max(Time.deltaTime, 1e-6f);
```

```
    // 速度がほぼ0のときは補正しない
    if (speed < 1e-3f)
    {
        prevPos = targetTransform.position;
        return;
    }

    // 3）必要な角速度（rad/s）= v / r
    float angVelRad = speed / Mathf.Max(radius, 1e-6f); // rad/s
    float angDegPerSec = angVelRad * Mathf.Rad2Deg;

    // 4）右回り（clockwise）なら負の角度（Unity の Y 回転は左回りが正）
    float sign = clockwise ? -1f : 1f;
    float angleThisFrame = sign * angDegPerSec * Time.deltaTime;

    // 5）プレイヤーの回転に適用（Y 軸回転）
    rig_target.transform.RotateAround(
        targetTransform.position,    // 回転の中心（A の座標）
        Vector3.up,                  // 回転軸（Y 軸回転）
        angleThisFrame// 毎フレームの回転量
    );
    GazeDataRecorder.Y_rotation += angleThisFrame;

    // 6）更新
    prevPos = targetTransform.position;
}

}
```

effect_manager.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System.Threading.Tasks;
```

```
public class effect_manager : MonoBehaviour
{
    public GameObject effectA;
    public GameObject effectB;
    public GameObject effectC;

    // Start is called before the first frame update
    void Start()
    {

        // effectA.SetActive(false);
    }

    // Update is called once per frame
    void Update()
    {

    }

    private bool eA = false;
    async public void EffectA()
    {
        //フラグ管理
        if (eA) return;
        eA = true;

        //視覚効果の発生開始を記録
        GazeDataRecorder.writeRed_point(1, 1);

        //点滅
        for (int i = 0; i < 4; i++)
        {
            effectA.SetActive(true);
            await Task.Delay(2);
            effectA.SetActive(false);
```

```
            await Task.Delay(80);

        }

        //視覚効果の終了を記録
        GazeDataRecorder.writeRed_point(1, 0);
        eA = false;
    }

    private bool eB = false;
    async public void EffectB()
    {
        //フラグ管理
        if (eB) return;
        eB = true;

        float b;
        Material mat;

        effectB.SetActive(false);
        effectB.SetActive(true);
        mat = effectB.GetComponent<Renderer>().material;
        b = mat.GetFloat("_BlurAmount");

        for (int i = 7; i > 0; i--)
        {

            //ブラーの強さを初期値に向けて変化させていく
            mat.SetFloat("_BlurAmount", b / ((((float)i * (float)i) + 9 * (float)i) /
10f));

            //視覚効果の発生開始を記録
            GazeDataRecorder.writeBlur(b, b / ((((float)i * (float)i) + 9 * (float)i) /
10f));

            await Task.Delay(100);
```

```
        }

        await Task.Delay(1200);
        for (int i = 1; i <= 7; i++)
        {
            //ブラーの強さを初期値に向けて変化させていく
            mat.SetFloat("_BlurAmount", b / ((((float)i * (float)i) + 9 * (float)i) /
10f));



            //視覚効果の終了を記録
            GazeDataRecorder.writeBlur(b, b / ((((float)i * (float)i) + 9 * (float)i) /
10f));

            await Task.Delay(150);
        }
        eB = false;

        mat.SetFloat("_BlurAmount", b);
        effectB.SetActive(false);
        GazeDataRecorder.writeBlur(b, 0);
    }

    private bool eC = false;
    async public void EffectC()
    {
        //フラグ管理
        if (eC) return;
        eC = true;

        float b;
        Material mat;

        effectC.SetActive(false);
        effectC.SetActive(true);
        mat = effectC.GetComponent<Renderer>().material;
```

```
            b = mat.GetFloat("_Saturation");

            for (int i = 7; i > 0; i--)
            {
                //彩度を初期値に向けて変化させていく
                mat.SetFloat("_Saturation", b / ((((float)i * (float)i) + 9 * (float)i) / 10f));

                //視覚効果の発生開始を記録
                GazeDataRecorder.writeSaturation(b, b / ((((float)i * (float)i) + 9 *
(float)i) / 10f));

                await Task.Delay(100);
            }

            await Task.Delay(1200);
            for (int i = 1; i <= 7; i++)
            {
                //彩度を初期値に向けて変化させていく
                mat.SetFloat("_Saturation", b / ((((float)i * (float)i) + 9 * (float)i) / 10f));

                GazeDataRecorder.writeSaturation(b, b / ((((float)i * (float)i) + 9 *
(float)i) / 10f));

                await Task.Delay(150);
            }

            eC = false;

            mat.SetFloat("_Saturation", b);
            effectC.SetActive(false);

            //視覚効果の終了を記録
            GazeDataRecorder.writeSaturation(b, 0);
        }
}
```