視線によって引き起こされる注意が空間知覚と物体認知に及ぼす影響

コース　　物質科学コース

指導教員　ミケレット・ルジェロ

学籍番号　１７０１３９

氏名　岡嶋佑弥

第 1 章　序論

1.1　研究背景

私達は膨大な情報に囲まれて生活している。選択に迷う程の情報量である現在の情報化時代では、価値のある情報も有害である情報も手に入れることは簡単にできる。その中で、それらを様々な視点において正しい情報や自分にとって役に立つ情報を見つけていかなければならない。必要な情報を手にする力、それが注意であり、私達は情報を取り入れるときには注意をうまく使う必要がある。

人間の脳で処理される多くの情報は目から入ってくる情報だと言われている。そのため視覚は人間が存在する上で重要な器官であるということが言える。人間がどのようにものを知覚しているのか、その機能を理解することは重要である。本研究も特に多くの情報を取り入れる視覚に着目していくものとする。

視覚は外界から得た情報が脳に届けられることによって、感情に多くの影響を与えたり危険を知り避けたりすることができる。また、その時の人間の状態や現れている感情を読み取ったりすることができる。私達は生活する中で無意識のうちに見ることを使って様々な情報を伝達し脳の働きを高めている。私たちが日常的に計測している視力は、ほとんど動かないで測った値である。しかし、実際の生活の中ではものを見る時に身体を動かして見ていることも多い。その背景も様々であり変化のある空間の中でそれぞれに適応しながらものを見ている。見るものに焦点を合わせたり、向きを変えたりしながら、筋肉をバランス良く使う力が私たちの生きている中の様々な場面で使われている。

今日コンピュータ技術等の発達によってより高次元な情報の映像を視聴することができるようになった。高度な映像情報を誰もが容易に得ることができ、場所を選ばずいつでも映像を見ることができる。このような社会の中では、人間が視覚から受ける影響はさらに増加していくものと考えられる。視覚は今まで以上に重要で大きな役割を持つようになる。

外界から視覚によって得られた情報は脳へと伝えられ、脳の様々な領域で処理されていく。見ることに依存している私たちの視覚は、目から受け取るすべての情報を平等には処理しない。目的に関する情報の一部分だけを選択して処理する。これは、「視覚的注意」と呼ばれる。視覚的注意の仕組みは異なる認知機能が統合され複雑化していることからまだよくわかっていない。

視覚的注意は視野の中の空間位置やものを選択することであり、有効視野を変化させるものである。さらに中心視野では処理能力が高く周辺部に比べ高い解像度を有するため、普段私達は当たり前のように選択した対象を眼球運動などによって中心視野に移動してものを見ている。

視覚系は受け取る情報すべてを同時に処理できないため、選択した場所に継続的に注意を向けて場面全体の処理を行っている。そのため視覚的注意が空間位置のどの場所に向けられるのかを決める視覚的注意の仕組みを理解することは視覚の働きを明らかにする上で重要である。

まず、ここでは立体映像視の注視点をいくつかの観点から分類することを考え、VRを使った視覚探索を行って視覚情報を測定した。

1.2　視覚探索

視覚探索とは様々な視覚刺激の中から、特定の目標刺激を探すことである。日常生活においては、人込みの中で決まった人を探す、連絡帳の中から目的の番号を探すなどの行動が例として挙げられる。

たとえば、右に傾いた（右肩あがりの）赤い線分を探索する問題を考える。図 1-1(a)の場合、色の特徴が他と異なっているため、容易に探すことができる。しかし、図 1-1(b)の場合、色や向きなどの特徴が他と共有されているため、ターゲットを容易に探すことができない。

このように、刺激のもつ単一の特徴にのみ注目すればターゲットか否かを判別できるような視覚探索を特徴探索（Feature Search）という。一方、「色」と「上がる向き」のような複数の特徴を考慮しないとターゲットか否かを判断できないような探索を結合探索(Conjunction Search）という。

以上、2 つの探索メカニズムをまとめたものが、特徴統合理論である。ふたつの探索条件を用い、画面内に表示される刺激項目の数を段階的に変化させ、ターゲットを見つけるまでの反応時間を調べると、図 1-1(c)のような結果が得られる。

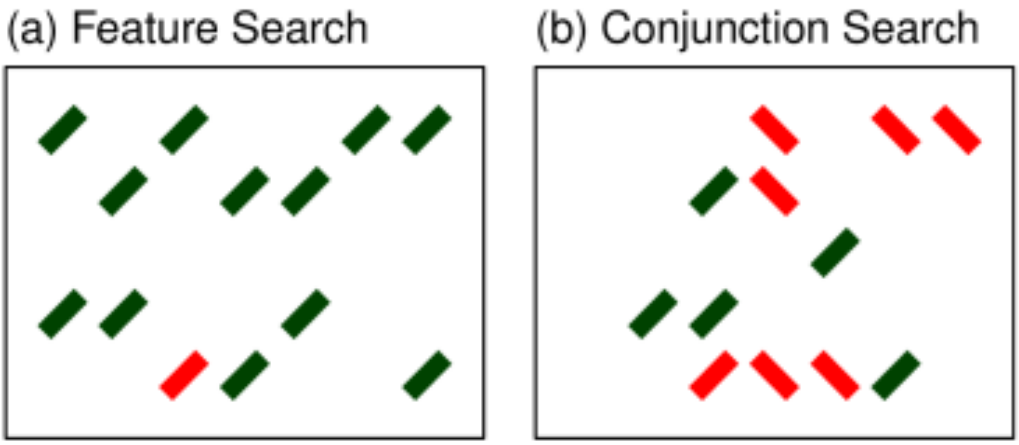
特徴探索の場合、グラフはほぼ水平になっていて、刺激項目の増加によって探索時間は変化しない（並列処理を行っている）。このような特徴探索における探索効率の特性をポップアウト効果（Pop-out Effect）という。一方、結合探索は複数の特徴に注目する必要があるため、刺激項目の増加にともなって、探索時間は増加する（逐次処理を行っている）。

以上から、妨害刺激の数と反応時間は関数関係にあることが分かる。（t：反応時間，n： 妨害刺激の数，c，d：定数）.

$$t = cn + d \quad \cdots \quad \text{式（1－1）}$$

探索関数の傾き c は、一つ一つの刺激を探すときの時間に相当する。図 1-1(a)のような探索画面では探索関数の傾き c は妨害刺激の数の影響を受けないことから、モニター上の刺激は並列的に探索され、図 1-1(b)のような探索画面では、探索関数の傾きcが妨害刺激の数に応じて線形に増加していることから、モニター上の刺激は逐次的に探索されていると考えられる。また、切片dは妨害刺激の個数によらない処理時間（キーボードを押すのに必要な運動時間など）を反映していると考えられる。

これらの時間（反応時間）の特徴から、視覚探索という心的処理には注意しなくてもターゲット刺激を見つけられる場合と、注意深く探索しなければターゲット刺激を見つけられない場合とがあると考えることができる。このように、目に見えない心的処理を、反応時間という指標を通して可視化することができる。



ターゲットの特徴が目に飛び込んでくる（並列処理）

色と形の特徴の組み合わせを一つずつ確認（逐次処理）

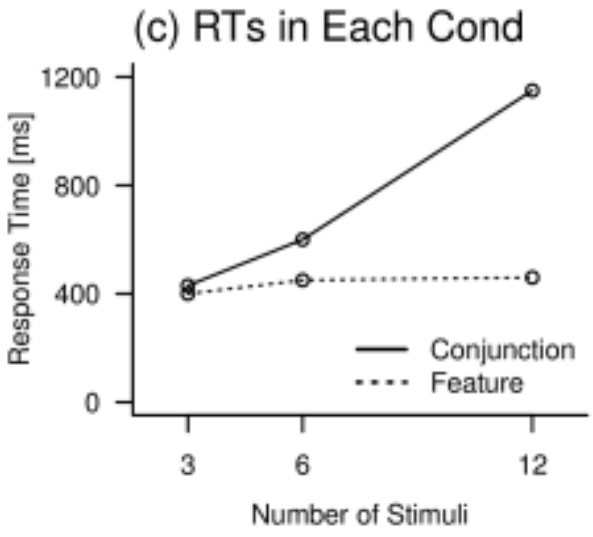


図 1-1　視覚探索課題の例とその反応時間[1]

[1] http://noucobi.com/neuro/RTanalysis/S1.html

2020 年 1 月 19 日最終閲覧[1]

1.3　先行研究

視覚探索における探索方略に関する先行研究を取り上げる。先行研究では、Search Panel 10 秒・Target Panel 3 秒・Break Panel 2 秒の計 15 秒 1 セットとした視覚探索課題を 10 セット行い、視覚探索のメカニズムを解明するための実験が行われている。

探索時のパネル上の視線重心（Center of mass）を計算すると(9.31, 9.47)となり、右上の方向に偏りがあることが分かっている。

また、探索方略を 4 つのパターンに分類し、マイクロサッケード運動にばらつきがあることが分かっている。

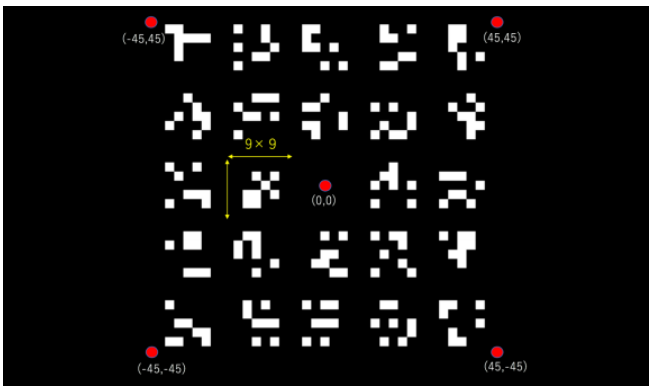


図 1-2　先行研究におけるパネルスクリーンのスケール[2]

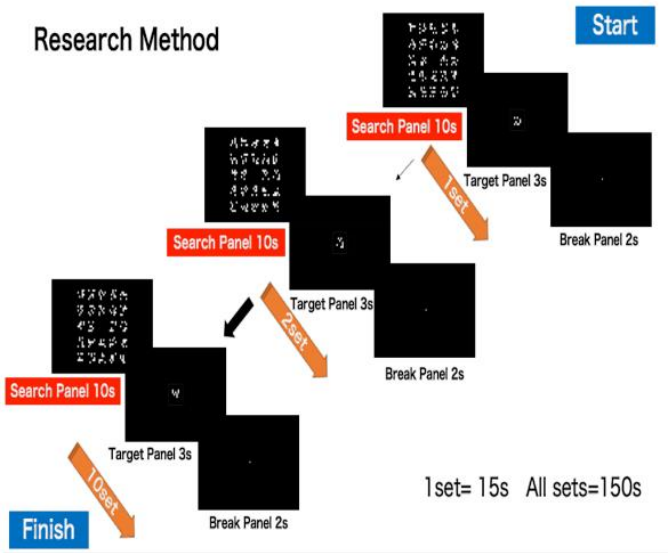


図 1-3　先行研究での実験の流れ [2]

[2] http://ruggero.sci.yokohama-cu.ac.jp/data/Maekawa_degree.pdf　から引用

1.4　研究目的

注意によって正答率が上がったり、刺激に敏感に反応できるようになる。ものがよく見えるようになり、また反応時間が短くなってより早く応答できるようになる。刺激に注意を向けているときは行動結果が向上すると共に脳の神経細胞の活動が増大する。また時間が有限であることから、時間認識に影響を与える要因とされている視覚を理解することにより時間を有効活用することができる。

このように注意は様々な利点との関わりが大きい。視覚は脳と感情を刺激し影響を与えるため、脳科学・心理学においても重要性は高いと思われる。注意を利用して伝えたい情報を効率良く伝える方法を見つける手段、また注意欠陥多動性障害（ADHD）のような発達・行動・障害の原因解明にも繋がると考える。日常的に取り入れることが容易である視覚情報が、身体への影響を引き起こすことにより感情を管理し、脳が活性化され記憶に働きかけることから学習面の向上が考えられる。また、感情を刺激し影響を与えることから痴呆症や鬱病の改善に期待できる。視覚的注意を理解することで様々な場面に応用することができる。

そのため人間がどのような仕組みで注意をコントロールしているのかを理解することは重要である。本論文では、VR を使い立体図形を注視し解答するという課題を用いて、定量的な視覚情報を得ることを目的とし、それによって視覚探索を行っている中でのユーザーごとに異なる視覚感性の特徴解析と視覚探索パターンの規則性を確認する。

先行研究ではパネルに表示されるオブジェクトは 2 次元であるため、本研究ではオブジェクトを 3 次元空間に拡張し、また、Target Panel と Search Panel の順番を入れ替えて実験を行った。

また、先行研究では反応時間について詳しく分析されていなかったので、本研究では反応時間も分析の対象とした。

第 2 章　実験概要

2.1　VR について

ここでは、近年急速に発展してきているバーチャルリアリティ（VR）について述べる。VR とは実際には存在しない人工的空間を用いて現実とは変わらない体験を実現させることを目指す技術である。ここで、そもそも「現実」とは何かと思うかもしれないが、VR がとる立場としては、「現実」とは五感などの感覚情報をもとに、脳が作り出した感覚体験である。人の脳は、全身の感覚器官から受け取った情報を統合することで外界を認識する。VR は入力（人がコンピュータに情報を入力する）とシミュレーション（入力情報に合わせて VR 環境がどう変化するかコンピュータが計算する）と出力（人の五感センサや脳に計算から得られた刺激・感覚を提示する）の三つの要素によって構成されている。実際に利用者は身体に装着する機器やコンピュータにより合成した映像や音響などの効果によって、3 次元空間内に自身の身体を投影し、空間への没入感を感じることができ、仮想現実を体験することができる。近年ではヘッドマウントディスプレイ（HMD）を用いることで、小型のディスプレーにコンピュータが作り出す映像を表示し、仮想現実の体験が可能になり、医療や教育、観光産業など幅広い分野で注目されている。

HMD とは VR 空間に入るために装着するゴーグルのような形状の、頭部に装着して使用する装置である（図 2-3 参照）。頭部に装着し、固定すると眼前にコンピュータなどから送られてきた像が投影され、映像を見ることができる。両眼で立体視させるものや、ハーフミラーで実際の風景と映像を重ね合わせるものもある。

今回の実験では、高性能の HMD である FOVE 社製の FOVE 0 を装置として使用した。本研究では現実世界に忠実な環境で実験を行うことを目的としており、そのために広い視野角や高解像度な表示が重要である。この装置は赤外線アイトラッキングシステムを用いているためトラッキング精度が高く、左右の眼 2 枚のパネルで構成されており、解像度も WQHD（2560×1440pixel）と高解像である。

VR と類似した技術として AR や MR という技術がある。AR（拡張現実）は Augmented Reality の略で、実際の画像や映像と CG の映像を合成することで、現実感のある仮想空間を作り出すことができる技術で、MR（複合現実）は Mixed Reality の略であり、CG で人工的に作られた仮想世界と現実世界を融合させる技術である。VR は現実世界とは全く切り離された仮想の世界を体験することが目的であることに対し、AR はあくまで現実世界に対して情報を付与することが目的である点で異なる。MR は VR と AR を融合させたものであり、現実と仮想の空間が融合し、仮想世界の情報を現実世界と重ね合わせて体験することができる。

そのため MR はバーチャルな世界をよりリアルに感じることができるため、ゲームや

エンターテイメントからビジネスシーンに至るまで、あらゆる分野で今後期待される技術である。

また、SR（代替現実）という新しい概念も存在する。SR は Substitutional Reality の略で現実空間とよく似た過去の映像を見せ、それを現在起きている出来事として錯覚させるものである。PTSD（心的外傷後ストレス障害）治療などの医療現場などに活用できるのではないかと考えられている。

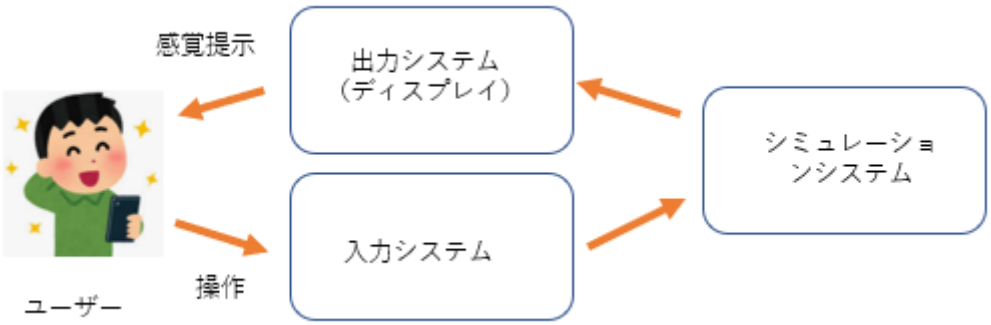


図 2-1　VR 生成のための基本的要素

2.2　Unity３D について

本実験で被験者が実際に体験する 3 次元空間は Unity3D というゲームエンジンを用いて作成した。Unity３D は主にコンピュータゲームや携帯ゲームを開発するために用いられている。このゲームエンジン Unity3D 自体は C 言語・C++で開発されているが、スクリプト言語としては C#、JavaScript 等を用いることができるが、本実験環境は C# 言語を用いてプログラムしている。

Unity３D というゲームエンジンを知覚実験に用いたのはその機能性にある。FOVE０はこのゲームエンジンプラグインに対応しているため、視覚探索を行っている被験者のヘッドデータおよび視線データをトラッキングし、プログラミングを用いて管理することができる。他に互換性のある VR 装置としては Oculus Quest 等があげられる。今回の実験で抽出したかったデータをエンジン上に保存し、記録できるようにプログラミングできる利点もある。現在までにこのゲームエンジンを用いて知覚実験を行った科学者は数少ない。

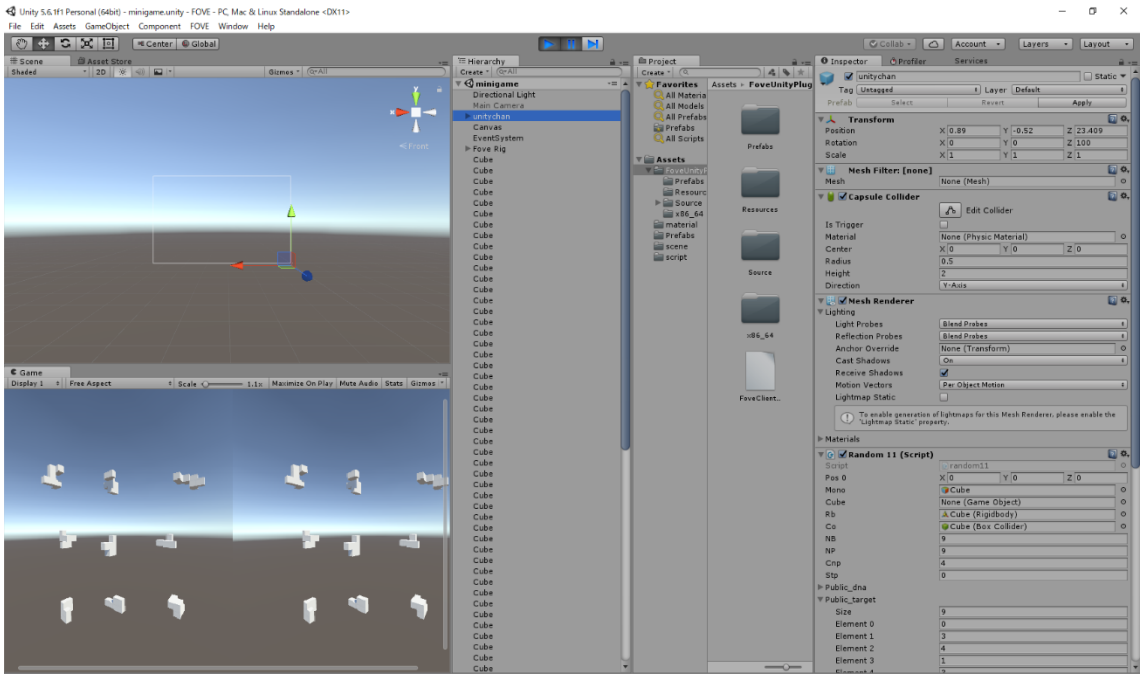


図 2-2　Unity３D 制御画面

2.3　実験環境

2.3.1　被験者について

被験者は、男女 25 名で全員晴眼者（眼鏡、コンタクトレンズ等の矯正も含む）である。被験者には「実験説明書」（p30 参照）を熟読してもらい、口頭説明による思い込み勘違いを防ぎ、それぞれの被験者に対して平等な説明をして実験操作を行えるようにした。被験者には実験の目的を伝えず、実験は、全て横浜市立大学にある理学系研究棟内の一室で行った。さらに静かな状態で実験に集中できるように、被験者に対しては必要以上に話しかけないようにした。

2.3.2　実験構成

本実験では視線追跡（eye tracking）を行うため、FOVE 0 を用いた。画面全てにピントが合っているという現在のスタンダードな VR　HMD は不自然で現実との差異を生み、画面の情報量が多いため、VR 酔いにつながっている。FOVE 0 は視線追跡技術によって、見ているところのみピントを合わせることができ、そのほかの領域を自然にぼかすことでより現実世界に近い VR 空間を作り出すことができる。目の開閉状況・瞬き・片目つぶり・奥行き情報などを含んだ視線まで把握することができる。また、目で見たところに照準が動くことによって大きく頭を振る必要がなく、三半規管にやさしいため酔いづらいと言われている。

本実験では各被験者がどこを見ているか正確に知るために、被験者が 1 点または一連の点を見るキャリブレーションを事前に行った。そうすることで、瞬き・頭の角度(x, y, z)・左右の眼の座標(x, y, z)の誤差を減らしている。また、被験者の実験はじめの戸惑いを減らし、投入感を損ねることなくスムーズに VR 空間に入ることができる。

解析において、より詳細なデータを得るためにフレームレートの数が重要であるが、Unity と同時起動であっても本実験では FPS70 で測定をすることができた。

なお、本実験は Unity 上の時間によって進められるため、時間が経過すると自動的に表示されるパネルが切り替わるようになっている。詳細は 2.3.3.実験方法で述べる。
アイトラッキングを用いた分析では、一般的に観測したデータから注視に関する情報を抽出し、近赤外線の角膜反射のパターンから、左右それぞれの目の注視点を推定することができる。

図 2-3　実験中の被験者の様子

2.3.3　実験方法

本実験では、視覚探索課題をすべて C#を用いたプログラミングにより作成した。

まず、被験者が VR 空間に慣れるためにプラクティスパネルを用意した。本実験ではメモライズパネルとして 1 つのランダムに作成された定量的な 3 次元立体が 7 秒間呈示される。次にサーチパネルとして 9 つの 3 次元立体が呈示される。被験者はメモライズパネルで覚えた立体がサーチパネルにあったときは左のコントローラーキーを押し、なかったときは右のコントローラーキーを押す。サーチパネルの最大表示時間は 15 秒としている。被験者ごとに試行回数を一定にするために問題数は 50 問とした。

また、いずれのパネルも鉛直方向（unity 上では*y*軸方向）を中心に回転するようにプログラムしている。

なお、本実験で FOVE 0 が取得しているヘッドトラッキングデータ・アイトラッキングデータ・反応時間・正誤パターンは解析がしやすいよう、CSV ファイルで記録している。

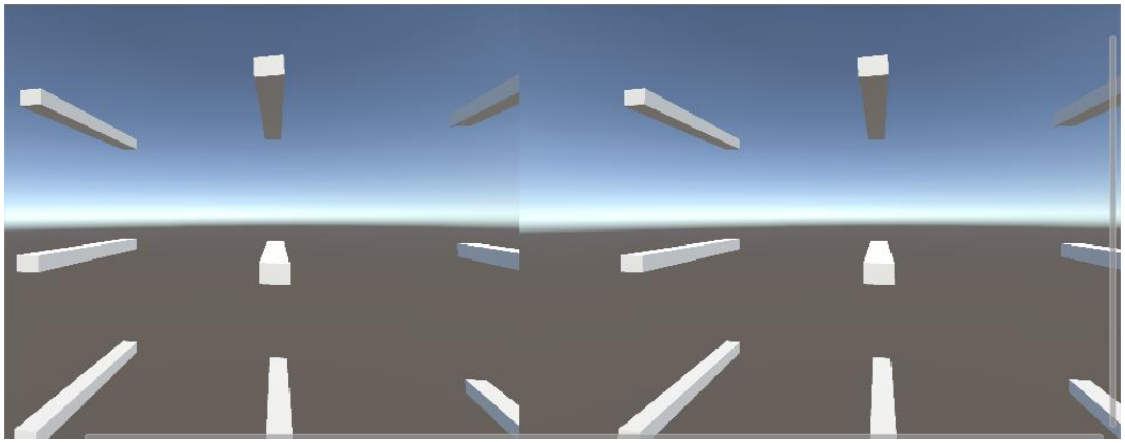


図 2-4　プラクティスパネル

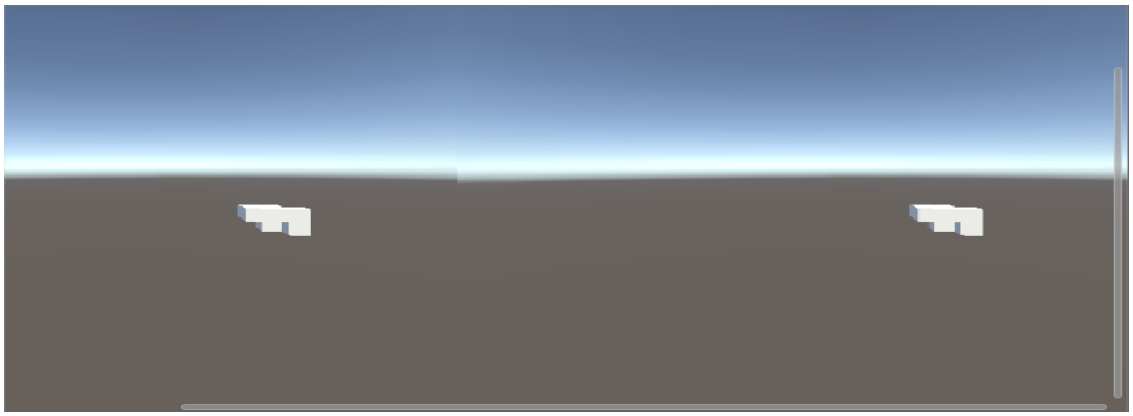


図 2-5　メモライズパネル

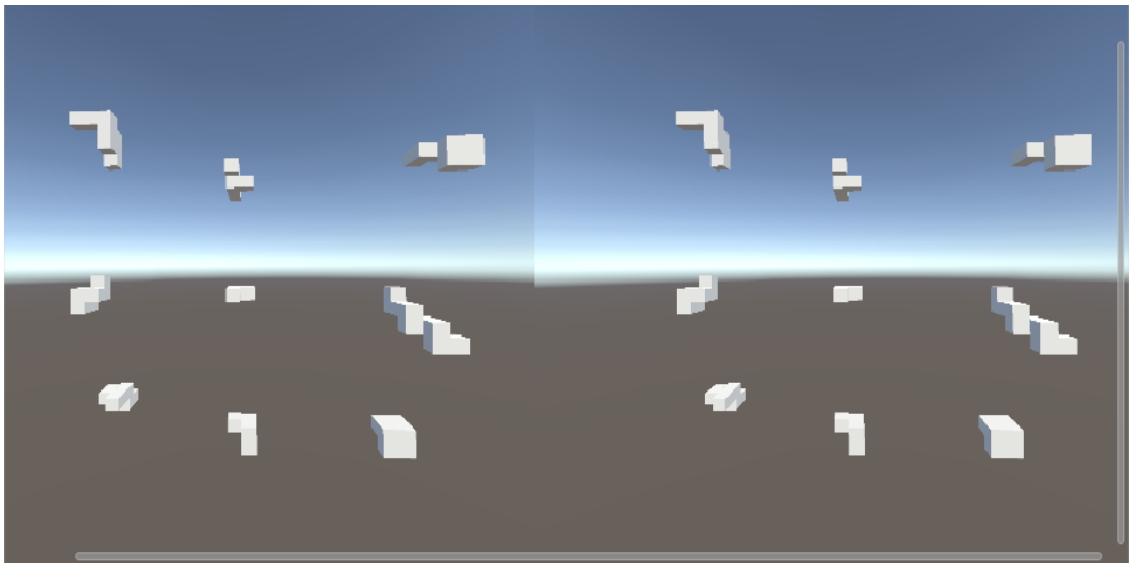


図 2-6　サーチパネル

第 3 章　実験結果・考察

3.1　視線情報の分布

3.1.1　角度の表し方

角度の表し方は主に 3 種類（クォータニオン, ロール・ピッチ・ヨー, オイラー角）ある。まず、「クォータニオン」について述べる。クォータニオンは複素数を 2 次元から 4 次元にまで拡張した複素数の事で 1 つの実部と 3 つの虚部からなる。クォータニオンを用いた回転について結論から先に述べると、絶対値($\sqrt{\omega^2+x^2+y^2+z^2}$) が 1 のクォータニオンは 4 次元の回転を表すことができる。

実際どのようになるか例を挙げる。クォータニオン $\omega+ix+jy+kz$ に対して、回転を表すクォータニオン $cos\theta+isin\theta$ を左からかけると

$$\begin{aligned}&(\boldsymbol{cos\theta+isin\theta})(\boldsymbol{\omega+ix+jy+kz})\\&=(\boldsymbol{\omega cos\theta-xsin\theta})+\boldsymbol{i}(\boldsymbol{wsin\theta+xcos\theta})+\\&\quad\boldsymbol{j}(\boldsymbol{ycos\theta-zsin\theta})+\boldsymbol{k}(\boldsymbol{ysin\theta+zcos\theta})\end{aligned}\quad \text{・・式（3-1）}$$

式（3-1）は次の座標変換を表している。

$$\begin{pmatrix}\boldsymbol{\omega}\\\boldsymbol{i}\\\boldsymbol{j}\\\boldsymbol{k}\end{pmatrix}\xrightarrow{\boldsymbol{cos\theta+isin\theta}}\begin{pmatrix}\boldsymbol{\omega cos\theta-xsin\theta}\\\boldsymbol{wsin\theta+xcos\theta}\\\boldsymbol{cos\theta-zsin\theta}\\\boldsymbol{ysin\theta+zcos\theta}\end{pmatrix}\text{・・式（3-2）}$$

ここで第 1 項・第 2 項及び第 3 項・第 4 項についてそれぞれ見ると、それぞれが 2 次元における角度θの回転式になっているため、クォータニオン$cos\theta+isin\theta$はωx平面とyz平面をそれぞれθだけ回転したものになっている。また、この計算結果には 1 点で交わる 2 つの平面上で同じ大きさの角度だけ回転させるというクォータニオンによる回転の重要な性質が現れている。

次に「ロール・ピッチ・ヨー」について述べる。これは、まず回転させたいオブジェクトにロール軸・ピッチ軸・ヨー軸という 3 つの直交軸を右手系になるように定義する。これらの軸を中心に回転させることをロール・ピッチ・ヨーといい、その角度をロール角・ピッチ角・ヨー角という。これらの軸は回転させるオブジェクトに固定されており、オブジェクトの回転とともに回転する。ここで、ロール・ピッチ・ヨーの回転角をそれぞれr, p, y とし、ロール軸・ピッチ軸・ヨー軸の方向ベクトルを中心とした回転行列を$Rr(\theta), Rp(\varphi), Ry(\psi)$　とすると、pを回転行列により回転した結果p'は式 3-3 のように表現できる。

$$\boldsymbol{p' = Rr(\theta)Rp(\varphi)Rx(r)} \text{・・式} \boldsymbol{(3-3)}$$

ロール・ピッチ・ヨーでは、ロール軸・ピッチ軸・ヨー軸をどうとっているか、それらの軸を中心にどの順番で回転させるかを決めなければならない。

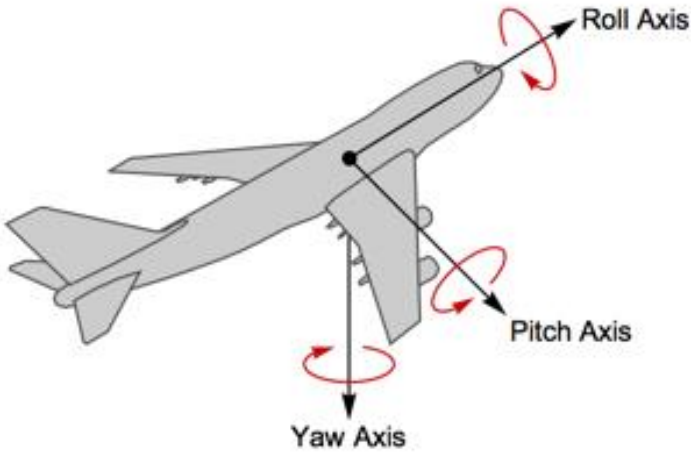


図 3-1　ロール・ピッチ・ヨー[3]

[3] https://qiita.com/yama04070319/items/d007af8e80c6c84f2de1
2020 年 1 月 19 日最終閲覧

最後に、「オイラー角」について述べる。オイラー角は剛体に固定された慣性主軸を表す座標系 (X, Y, Z) と慣性系 (x, y, z) を結び付ける角度である。z軸とZ軸のなす角度を β 、β が 0° または 180° ではない場合には、xy平面とXY平面は一つの直線で交わる。この交線を N、x 軸と交線 N のなす角度を α とし、X軸と交線 N のなす角度を γ としたとき、(α, β, γ) がオイラー角である。
本実験では、具体的な姿勢をイメージしやすく、パラメーター数も 3 つと処理がしやすい「オイラー角」を用いた。

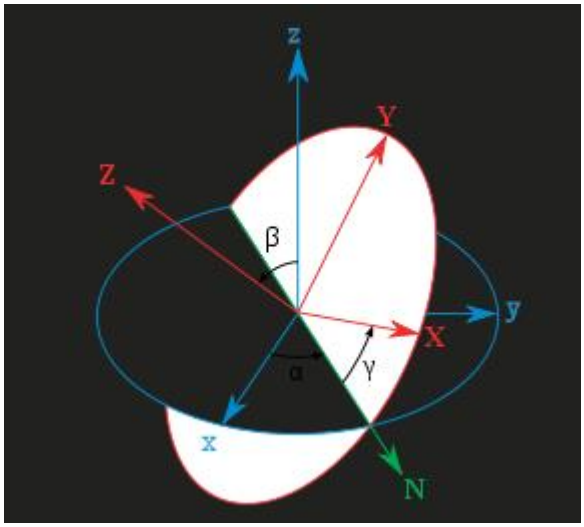


図 3-2　オイラー角[4]

[4] https://www.mauriciopoppe.com/notes/computer-graphics/transformation-matrices/rotation/euler-angles/　2020 年 1 月 19 日最終閲覧

3.1.2 視線情報のヒートマップ

本実験では、視線の角度（オイラー角）を視線情報として被験者別に取得した。以下、directionX を水平方向の傾き、directionY を鉛直方向の傾きとする。得られた視線情報を用いて Python プログラムで解析を行った。

図 3-3 はサーチパネルが表示されている時間における視線角度を 50 問分足し合わせてプロットしたものである。この視線分布に偏りがあるのではないかと思い、プロット結果を視覚的にわかりやすい 2D ヒストグラムにした（図 3-4）。

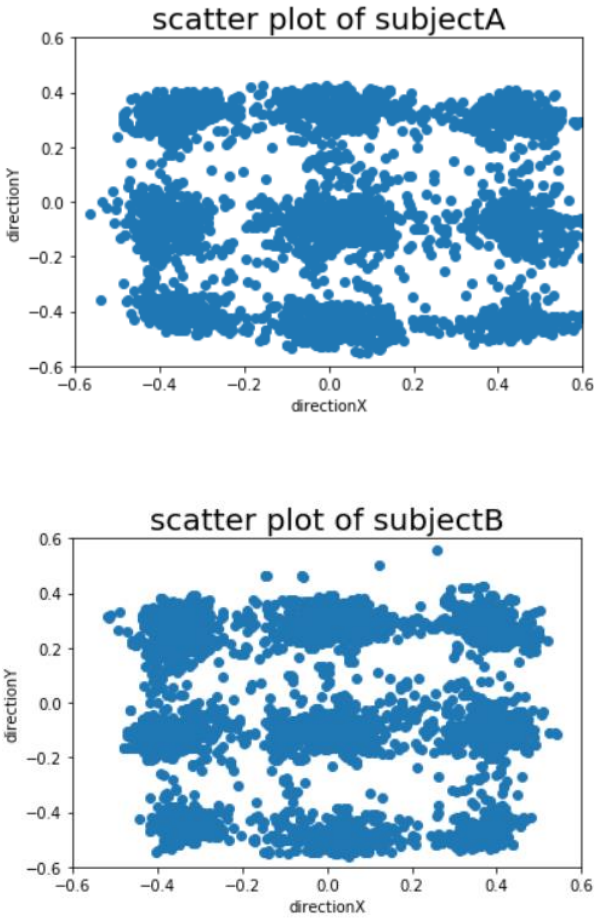


図 3-3 被験者別の視線情報の散布図

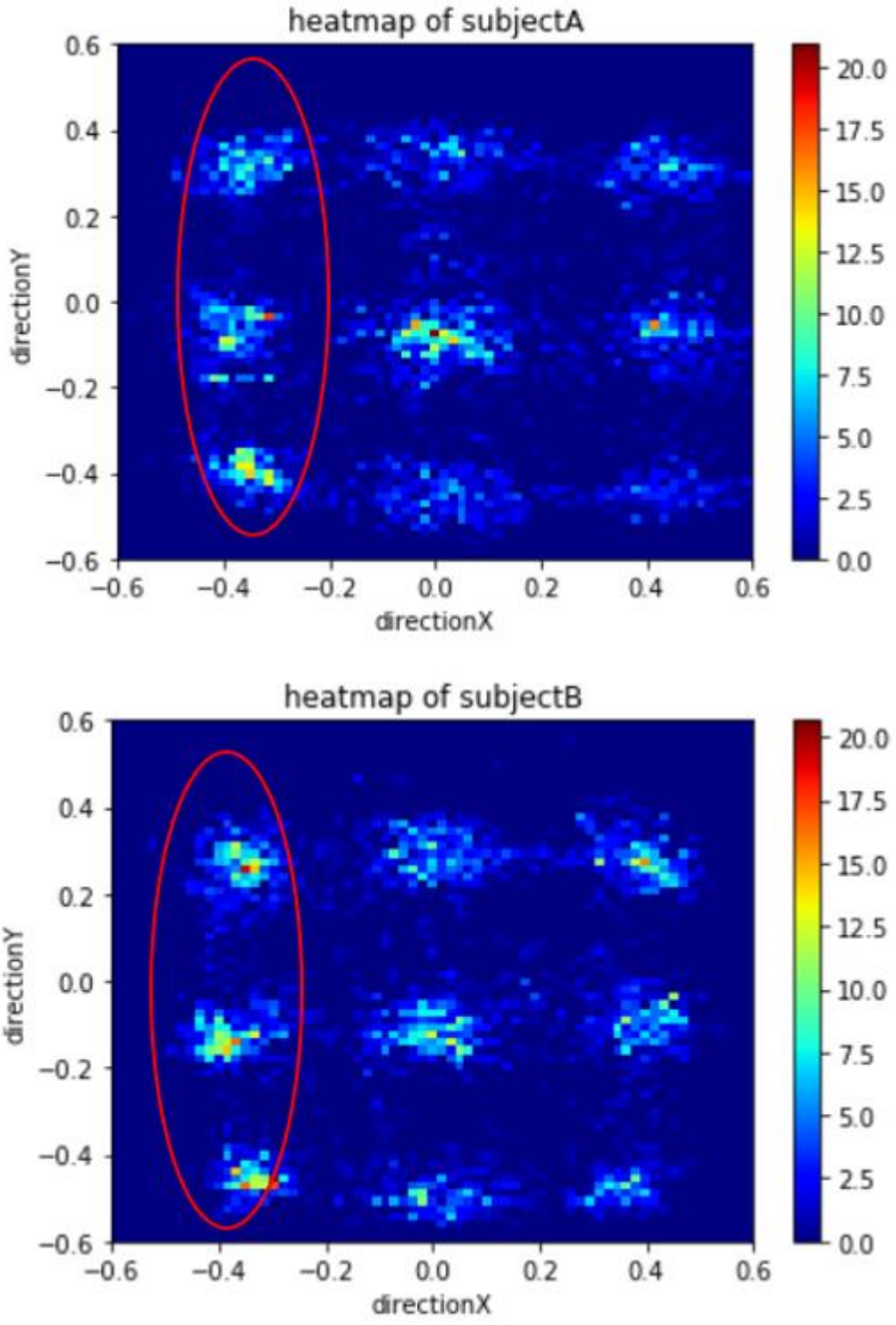


図 3-4　被験者別の視線情報　２D ヒストグラム

被験者 A、被験者 B の視線情報の２D ヒストグラムから視線は左側に分布しやすいのではないかと予測することができたため、被験者２５人全員の視線情報を統合し、新たな２D ヒストグラムを作成した（図 3-5）。

２D ヒストグラムは Python プログラムを用いて作成した。またグラフのx軸及びy軸の上限・下限、及び bins は全て同じ値を用いて解析を行った。

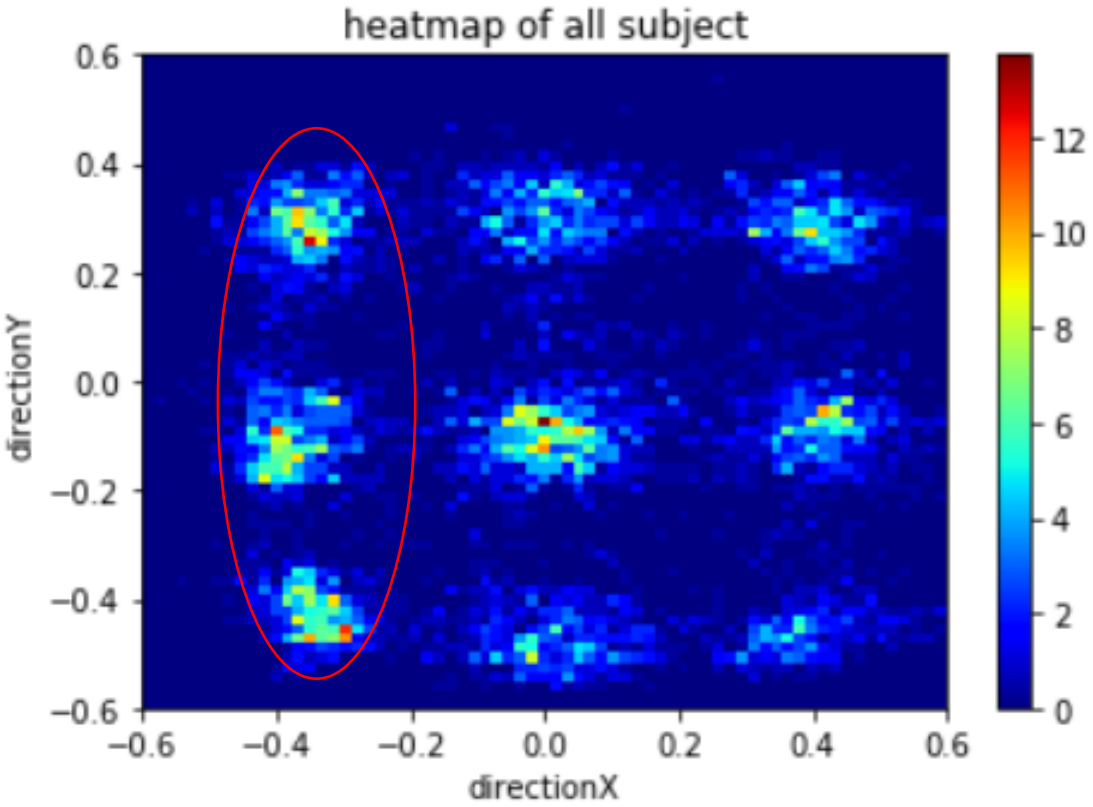


図 3-5　被験者全員の２D ヒストグラム

図 3-5 から、視線情報は左側に分布しやすい傾向があることが分かった。先行研究では視線重心が右上になる傾向が表れていたため、先行研究とは逆の結果を示すこととなった。これは先行研究ではターゲットが 2 次元で平面的であるため、左脳優位となり右側に視線が集中しやすく、本研究では、ターゲットが 3 次元で立体的であるため右脳優位となり左側に視線が集中しやすくなったのではないかと考えられる。

この結果は、本などの文章は左から書かれているためヒトの視線は左に偏りやすいことと関連があるのではないかと考えられる。また中心のヒートマップが濃くなっているのは、記憶時に表示される１つの３次元立体が中心の位置に存在していたことが関係していると考えられる。

3.2　視線の動き

3.2.1　正誤パターン

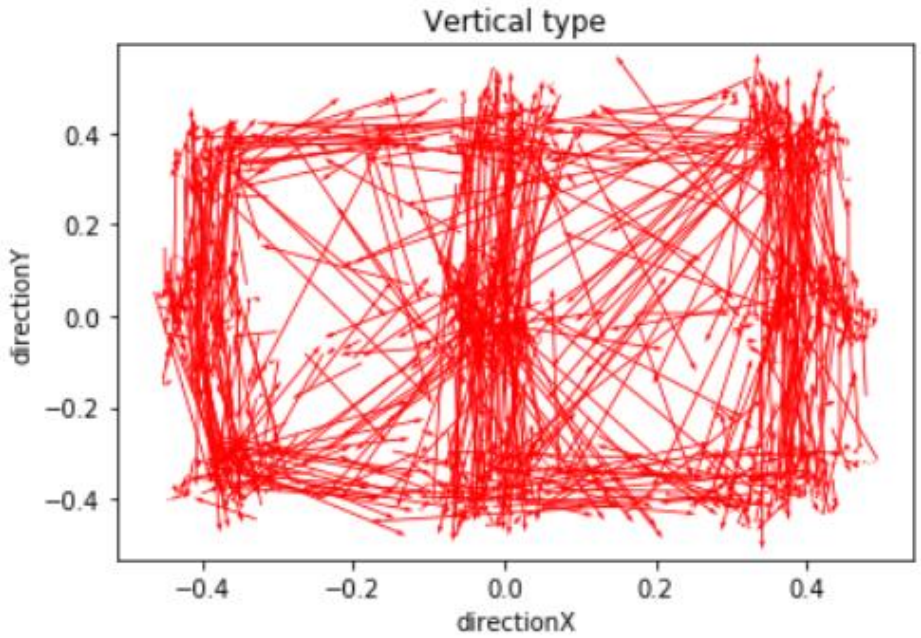
本実験では、メモライズパネルに表示される立体がサーチパネルに存在しているか否か、また課題に正答しているか否かの 2 つの指標を用いて 4 つのパターンに分類した（表 3-1）。これ以降、表の上段で用いたアルファベットを用いて正誤パターンを表記する。

表 3-1　実験の正誤パターン

ABS（ABSENT）	FA（FALSE　ALARM）	HIT	MISS
無かったものを無かったと回答	無かったものをあったと回答	あったものをあったと回答	あったものを無かったと回答

3.2.2　ベクトル解析

探索時の目線の動きを Python プログラムの plt.quiver()を用いて解析した。その結果視線の動き方は大きく 2 つに分けられた。（図 3-6）



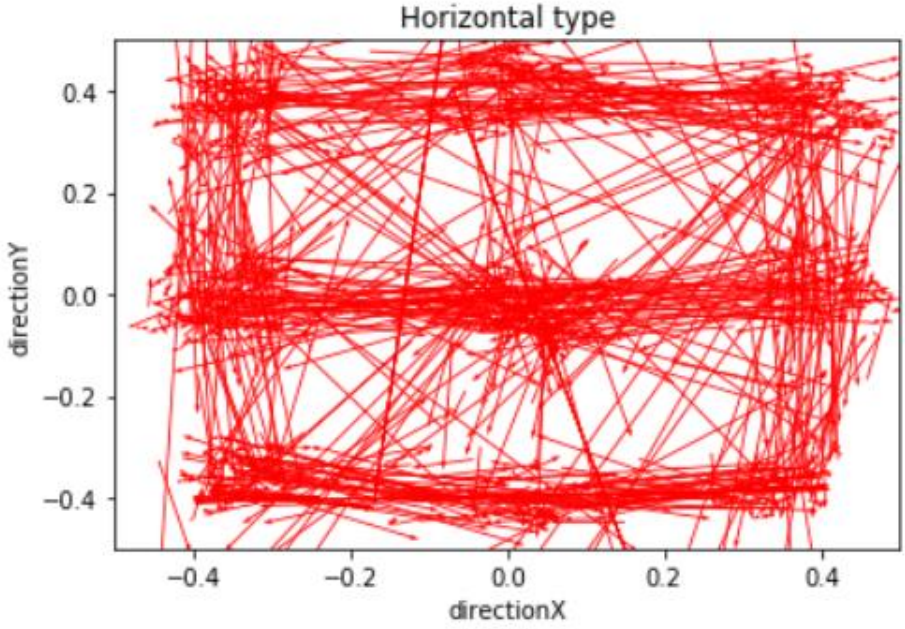


図 3-6　フレームごとの視線のベクトル

視覚探索において縦向きに視線を動かす傾向があるものを Vertical Type（垂直方向）、横向きに視線を動かす傾向があるものを Horizontal Type（水平方向）と呼ぶ。いずれにおいても斜めの動きは少ないことが分かる。また各パターンの正答率（ABS または HIT の割合）は Vertical Type が 72.3%であったのに対し、Horizontal Type では 80.4%であった。

この結果から探索時には視線を水平方向に動かすと正確に効率よく探し出せる可能性が高いと考えられる。

また、頭の揺れ幅が大きいと眼球運動をスムーズに行うことができず、正答率が下がると考えられたが、本実験ではその傾向は確認することができなかった。

3.3　フーリエ変換

フーリエ変換とはデータ解析手法の一つで一般的には時間領域のデータを周波数領域へと変換するためのアルゴリズムとして使用されている。

周波数ωの正弦波を時間関数$f(t)$と考えると、一般には以下の式になる。

$$f(t) = Aexp(iwt) \cdots \text{式}\ (3-4)$$

Aは振幅、iは虚数単位である。

expが正弦波になるのはオイラーの定理を用いて確認できる。正弦波は時間関数または周波数関数とみなすことができる。正弦波以外についても正弦波の直交性（正弦波は互いに線形独立である）と完全性（任意の関数は正弦波の和に分解することができる）という 2 つの性質から同様のことがいえる。

$$F(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(t) e^{-iwt}\, dt \cdots \text{式}\ (3-5)$$

時間tの関数$f(t)$を周波数ωの関数$F(\omega)$に移す変換の事をフーリエ変換といい、周波数ωの関数$F(\omega)$を時間tの関数$f(t)$に変換することを逆フーリエ変換という。

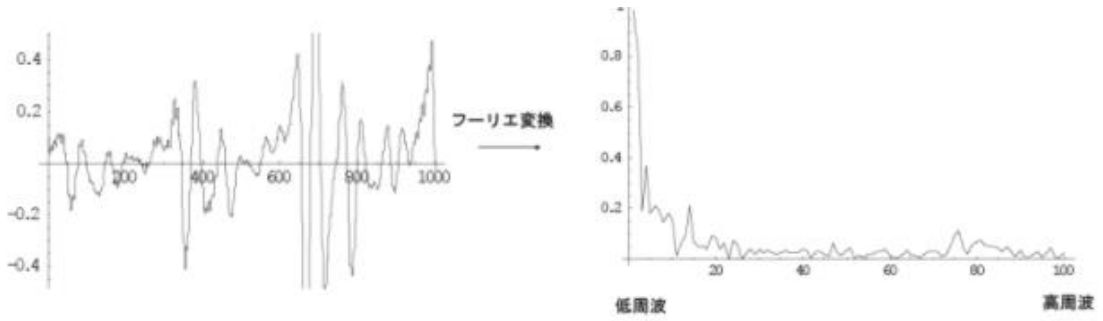


図 3-7　フーリエ変換の例[5]

[5] http://www.ne.senshu-u.ac.jp/~proj20-20/before/02.html

2020 年 1 月 21 日最終閲覧

3.3.1　フーリエ変換を用いた解析

１フレームごとの移動距離から視線ベクトルの速度を算出し、探索時の正誤パターン別速度の波形をフーリエ変換した。（図 3-8）

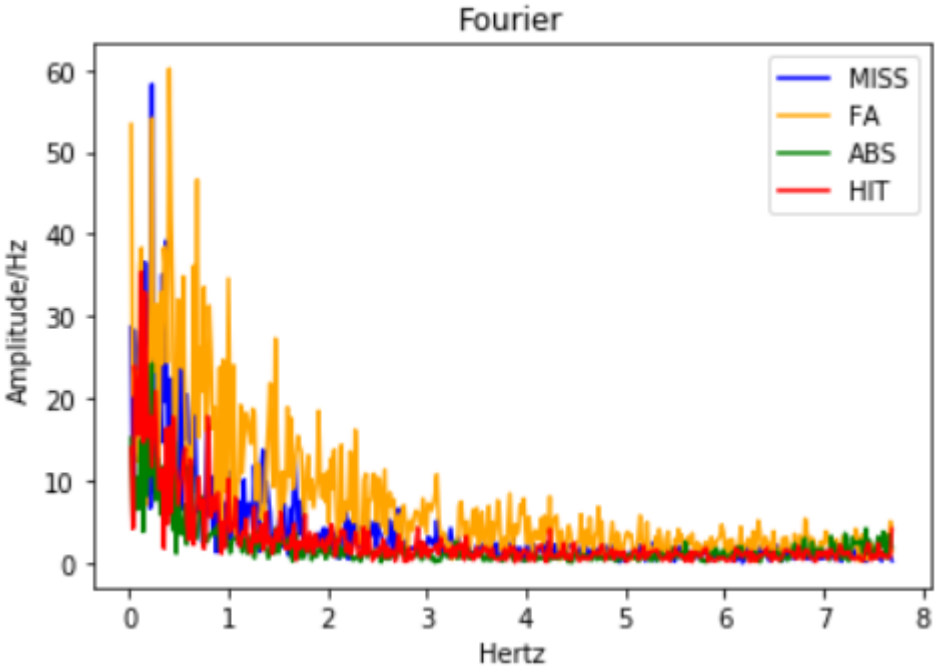


図 3-8 探索時の　のパターン別フーリエ変換

図 3-8 のグラフ結果から、正誤パターンいずれにおいても周波数 0~1Hz をピークにして収束していくようになったが、正誤パターンによって大きな変化はなく、本実験では視線の速度と正誤パターンに関連性は見られなかった。

3.4　反応時間

反応時間（Reaction　Time）とは、刺激が与えられてからその刺激に対して外的に観察可能な反応が生じるまでの時間である。反応時間は心理学研究において多く用いられる特徴量である。特に、ヒトが何らかの知覚課題を行う際の反応について言う。刺激とはモニターに映し出される図形など、被験者に呈示するものの事を指す。反応時間の測定には、刺激の呈示を制御する装置、反応を取得して記録する装置、刺激の呈示によって起動し反応の取得によって停止する時間計測装置の 3 つが必要である。

本実験ではプログラミングによって、刺激の呈示の制御と反応時間の計測、キーボードで反応を取得した。反応時間は刺激の入力から反応までに起こる様々な心的処理（刺激の知覚、判断や反応選択、反応の運動実行）が加算されたものとして考える。反応時間は課題遂行成績の重要な指標であり、反応時間が長いほど複数の心的処理が関与していると考えることができる。

反応時間には刺激を知覚したらボタンを押すのみの単純反応時間と、刺激を知覚したらその内容によって押すボタンを判断する選択的反応時間がある。

本実験では、サーチパネルが表示されてから課題を遂行しキーボードを押すまでの時間を反応時間としたため選択的反応時間を使用している。

3.4.1　パターン別反応時間の分析

パターン別に反応時間の平均を求めると図 3-9 のようになる。

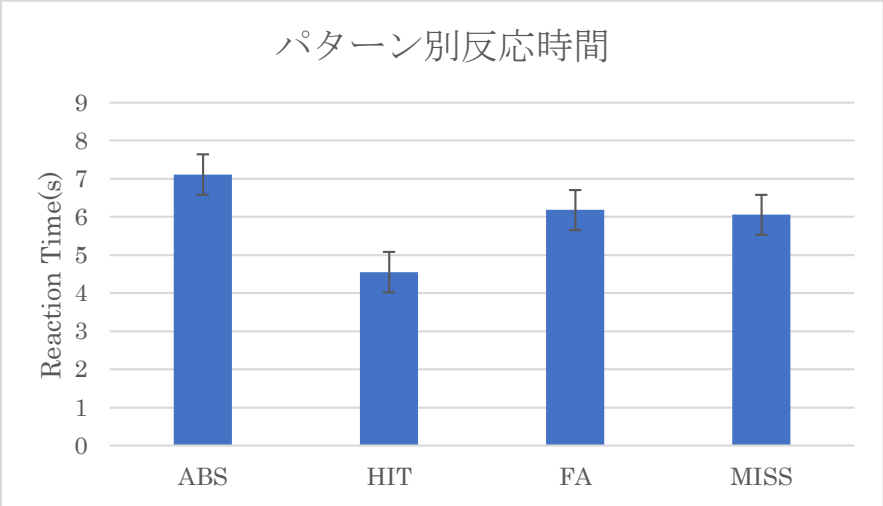


図 3-9 パターン別反応時間

この結果から、パターン別で反応時間の差に有意性があるのではないかと考え、この反応時間を使って統計的分析を行った。まずは t 検定と分散分析を用いて、実際に反応時間平均に差がある可能性が高いかどうかの確認を行った。

＜分散分析＞

分散分析を行った結果を表 3-2 に示す。

表 3-2　パターン別反応時間の分散分析結果

変動要因	変動	自由度	分散	観測された分散比	P-値	F 境界値
グループ間	204.6	3	68.22192	10.84637	1E-06	2.641

p 値からパターン間で平均反応時間に優位性があると考えられる。次に、どのパターンがこの結果に影響を与えているか調べるために、t 検定を用いた。

＜t 検定＞

今回は分散が等しくないと仮定した 2 標本による t 検定を用い、有意水準は 5%としている。表 3-2 の下段は p 値を表している。

表 3-3　パターン間の t 検定結果

ABS と HIT	ABS と FA	ABS と MISS	HIT と FA	HIT と MISS	FA と MISS
3.72E-11	0.127	0.0452	0.0107	0.00606	0.86

表 3-3 の結果から、HIT はいずれのパターンとも有意水準 5%で帰無仮説を棄却できることが分かった。

ここで、反応時間は正規分布に従わないという性質があるため、ほかのデータと違い、その解析方法に配慮する必要がある。正規分布に従わない原因としては、以下のようなことが考えられる。なるべく早く反応しようとするとき、反応時間は短くなり、分布は左に偏る。しかし、反応時間が負になることはなく、また筋の収縮にかかる時間などの不可避な時間を考えると、反応時間の短縮はある程度で頭打ちになる。

一方、長くなる分には時間は無限に長くなることができ、たくさんの試行のうちの少数の試行において、注意散漫やキーボードの押し間違いなどにより、時間が長くなってしまうことがある。

t 検定や分散分析は認知科学の研究で主にデータ分析方法として利用されるが、「型どおり」パラメトリック検定であるためそのデータの母集団が正規分布に従っていることを前提に検定を行っている。そのため、反応時間の生データにこれらの手法を適用することは望ましくない。その理由は、前提条件をみたしていないデータに対し検定を行うと、母集団の平均値に差がないのにも関わらず、差があると判断してしまうタイプ 1 のエラーを起こす可能性が高まるからである。また、正規分布に従わない反応時間の平均値など代表値を用いて群間の比較をすることには問題がある。そこで反応時間解析で用いられる主な手法として以下の 3 つがあげられる。

１．対数変換

変数変換（transformation of variable）とはデータに対して特定の変換を行うことで、分布の形状を変化させる手法である。分布の形状を変えたいので変換は非線形となる。反応時間解析においては、対数変換（logarithmic transformation）を用いることが多く、これにより多くの反応時間データが正規分布に近づくことが知られている。

しかしここで注意しなければならないのは、検定の結果から得られた有意差は変数変換後の値に対して保証されるものであって、元のデータにおいても差があるかとい

えるかどうかはわからないということである。

また、変換後の確率変数が必ずしも正規分布に従うかどうかは分からないため、変数変換を行っても正規分布にならず、パラメトリックな統計手法を適用できないこともある。図 3-10 では左が対数を取る前、右が対数を取った後の分布を表している。

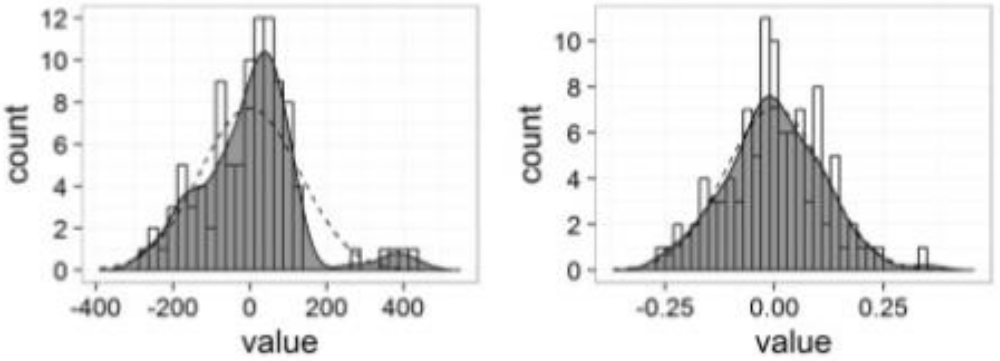


図 3-10　対数変換による分布の変化[6]

２．分布のフィッティング

反応時間は正規分布に従わないということを先に述べたが、反応時間がよく従う分布として、指数分布と正規分布を掛け合わせた ex-gaussian 分布が知られている。この分布に反応時間データをフィッティングさせることで分布同士の平均値などパラメータを比較することができる。問題点としては、この分布に反応時間が従うという合理的根拠が見つかっていないということである。

３．正規分布に従わない統計的手法

対数変換や分布のフィッティング以外に、正規分布に従わない統計的手法として、解析の対象データ（今回ではパターン別反応時間）に対して、一切の分布を仮定しないノンパラメトリック検定がある。ノンパラメトリック検定には対応のある 2 群の「マイ・ホイットニーの U 検定」、対応のある 2 群の「ウィルコクソンの符号順位和検定」、対応のない 3 郡以上の「クラスカル・ウォリス検定」、対応のある 3 郡以上の「フリードマン検定」などが挙げられるが、本実験で用いた反応時間データは被験者一人ひとりからパターン別の反応時間を集めて分析しているため、対応ありのデータである。そこで、ウィルコクソンの符号順位和検定を用いて、HIT と他のパターンの平均に差があると言えるかどうかを分析した。

---

[6] https://www.slideshare.net/itoyan110/chapter7-94652905　2020 年 1 月 19 日最終閲覧

2 つの測定値間の各差Z_iの絶対値を順番に並べ、その順位をR_iとし、$\emptyset_i$を指示関数とすると、以下のようにしてウィルコクソンの符号順位統計量を計算することができる。

$$\boldsymbol{W = \sum_{i=1}^{n} \emptyset_i R_i \quad \cdot\cdot\cdot 式\ (3-6)}$$

計算結果は以下のようになった。(表 3-4)

表 3-4　HIT と他のパターンのウィルコクソンの符号順位和検定結果

HIT と ABS	HIT と FA	HIT と MISS
3.124E-10	0.0050	0.0104

反応時間の統計的分析から HIT は他の正誤パターンに対して有意性を持つことが分かった。

そのため、HIT のときの、脳波などの解析を行う事によって、高齢者の反応時間低下のメカニズムを解明、障害物からよける、車両事故を減らすことにつながるのではないかと考えられる。

第 4 章　まとめと議論

これまで立体映像視における注視点について考察をしてきた。本研究では大きく分けて 3 つの結果を得ることができた。1 つ目は実験前の仮説通り、探索時の視線ベクトルにおいて、Vertical Type と Horizontal Type の 2 種類に大別できることが確認された。また、それぞれの正答率から Horizontal Type のほうが効果的に探索を行えることが分かった。2 つ目は、4 種類の正誤パターンにおいて、HIT 時の反応時間の有意性を示すことができた。これらの特定の知覚パターンの発見は、人間の知覚に関連する新しいデバイスまたはシステムの設計にとって重要である。たとえば、被験者の視線追跡を継続的に監視することは、被験者の知覚パフォーマンスを制御及び最適化するのに役立つ。3 つ目は、被験者全員の探索時における視線分布の解析から、仮説と反して視線は左側に寄りやすい傾向があることの有用性を示すことができた。これは 3 次元の立体視により右脳が活躍したものからであると考察した。このことは VR で確認されており、特定の分野に応用することが可能である。たとえば、軍事、機械設計、その他の実世界の 3 次元デバイスでは、私たちの発見が応用される。

注意機能は高次脳機能の基本的な能力の一つであり、注意障害に陥ると、外界の刺激への反応の衰え・集中力の低下など日常生活に影響がでる。また、転倒は運動機能だけではなく注意分散能力である認知機能にも原因があると考えられる。立体図形の認識には、注意持続性・視覚探索力・視覚運動性などのさまざまな能力が必要である。注意機能を知ることは、複数の認知機能の維持・向上につながるのではないかと期待する。

今後の展望としては、本研究では立体映像の中心視野における物体知覚を行っているが、注意によって選択された対象以外の範囲部分である周辺視における視覚の役割もあると考えられていることから、周辺視野を対象とし解析する必要があると考える。また、人間の特性を理解するため様々な行動をデータで取得し分析することが必要である。感情を刺激する形状・色彩などが思考手段や情報処理、また問題解決方法に与える影響も探っていきたい。

第 5 章　謝辞

本研究を進めていくにあたり、ミケレット・ルジェロ教授には、実験方法、分析方法など研究や発表ついての熱心なご指導を賜りました。心より感謝いたします。

また、ゼミや日々の研究においてさまざまなアドバイスをくださったミケレット研究室の方々にも深く感謝いたします。

そして、本研究における実験の際に被験者を快く引き受けてくださった皆様に感謝いたします。

最後に、本論文執筆に携わってくださった全ての方々に、感謝の意を示し謝辞といたします。

第 6 章　参考文献

Miller, M.R., Herrera, F., Jun, H., Landy J.A. & Bailenson, J.N.(2020). Personal identifiability of user tracking data during observation of 360-degree VR video

Yuichi, Itoh.,Ehud,Sharlin.,Yoshifumi, Kitamura., Fumio, Kishino., Benjamin, Watson., Steve, Sutphen., & Lili, Liu.(2006) A user interface for spatial cognitive assessment using ActiveCube

Nascimento, A.M., Queiroz, A.C.M., Vismari, L.F., Bailenson, J.N., Cugnasca, P.S, Camargo, J.B., & de Almeida, J.R. (2019) The role of virtual reality in autonomous vehicles' safety

Koji, Nishino., Hiroko, Irishio., Hiromitsu, Yokohama., Yukimasa, Miyagishi. (2017) Individual difference in using spatial signs cognitive characteristics in VR -The Difference between Vacant Lots and Landmarks-

Hodaka, Tasaki., Takumi, Hayashi., Kyoko, Hine. (2017) Active-passive view affects impression and memory in virtual reality, Tokyo Denki University

Takatsune, Kumada., Yasuo, Kuchinomachi., Shinya, Saida. (1995) The properties of functional visual fields on visual search : Evidence from manual reaction time and saccadic eye movement, National institute of bioscience and human-technology

Ryuta, Iseki. (2020) How have psychologists treated response times?

Rei, Maekawa. (2020) Research on search strategy in eye movement in visual search

Makoto, Ono. (2019) The evaluation of mental clock feelings using a three dimensional virtual reality environment

大山 正(2010),『知覚を測る~実験データで語る視覚心理学~』,株式会社誠信書房

沖中 大和, 満上 育久 八木 康史(2016),『人の眼球と頭部の協調運動を考慮した視線推定』 大阪大学

佐久田 博司, 平井 敬子 武士俣 貞助(2000)『3 次元モデルを用いた空間認識力改善と評価法』

舘 暲, 佐藤 誠, 廣瀬 通孝 (2010) バーチャルアリティ学 日本バーチャルリアリティ学会

反応時間解析の理論と応用
http://noucobi.com/neuro/RTanalysis/S1.html　2020 年 1 月 19 日最終閲覧

フーリエ変換おもろいわー！
http://www.ml.seikei.ac.jp/biolab/lecture/Bioelecronics/2.5Fourier.pdf　2020 年 1 月 19 日最終閲覧

知っておきたいキーワード反応時間
https://www.ite.or.jp/contents/keywords/2003keyword.pdf　2020 年 1 月 19 日最終閲覧

データ科学便覧
https://data-science.gr.jp　2020 年 1 月 19 日最終閲覧

反応時間解析の理論と応用
http://noucobi.com/neuro/RTanalysis/RTanalysis.html　2020 年 1 月 19 日最終閲覧

クウォータニオンと回転
https://www.f-sp.com/entry/2017/06/30/221124　2020 年 1 月 19 日最終閲覧

実験対象者の方へ

実験説明書

実験にご協力いただき誠にありがとうございます。この操作説明を読んで、実験の流れを理解していただいてから実験を行います。

＜実験方法＞

準備

１．椅子に座り、実験説明書をお読みください。
２．*キャリブレーションを行います。
３．画面に表示される指示に従って、目の位置を調節してください。緑の玉が表示されますので、それを追うようにして目を動かしてください
４．「調整が完了しました」が表示されたら教えてください。
５．キャリブレーション後は VR 装置に触らないようにしてください。

操作方法

１．表示される１つの３次元立体を記憶してください。
２．表示される９つの３次元立体の中に覚えた立体があった場合は左のコントローラーキーを押し、なかった場合は右のコントローラーキーを押してください。
３．１と２を繰り返し行ってください。
４．何度か練習を行います。
５．練習が終了したら実験を開始します。時間は 10 分程です。

＊キャリブレーションとは実験開始前に個人間による目の位置やズレを調整するためのものです。

注意事項

・実験中は手元の視界が失われるため、キーボードの位置が分からなくならないよう、手は動かさないようにお願いします。

・表示される立体のうち回転対称のものは同じものとして考えてください。

例

データの取り扱いについて

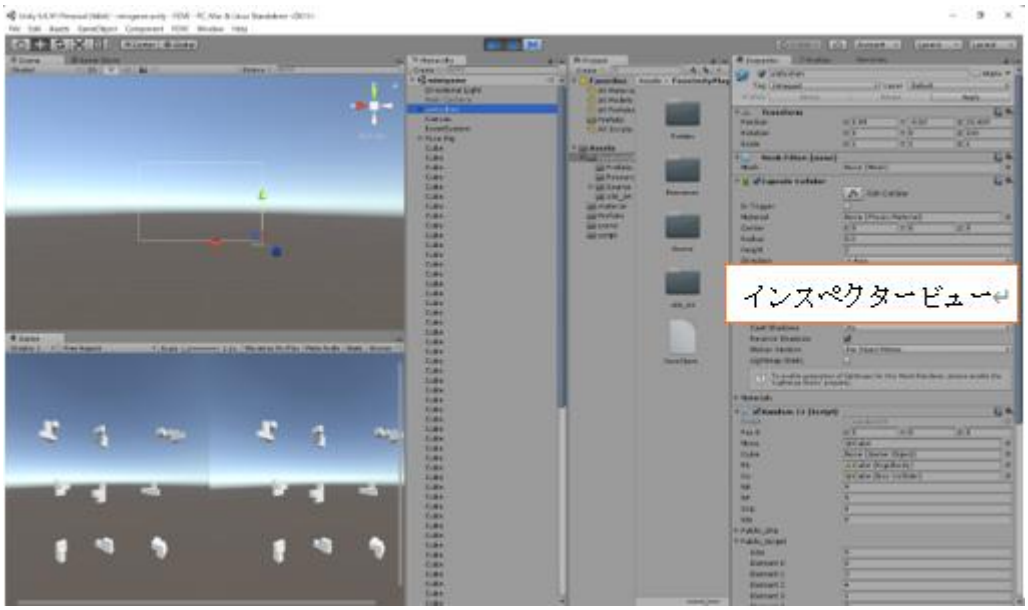
　取得したデータは本研究以外では使用しません。またデータは番号付けを行い匿名化し、ランダムに分析を行います。また、この実験は個人の能力差を測るものではありません。個人間で能力の優越を分析することは一切ありません。

横浜市立大学
国際総合科学部　国際総合科学科
理学系　物質科学コース
ミケレット研究室
岡嶋　佑弥

付録

本研究で用いた unity の実験環境及び C#のプログラムコードをまとめる。

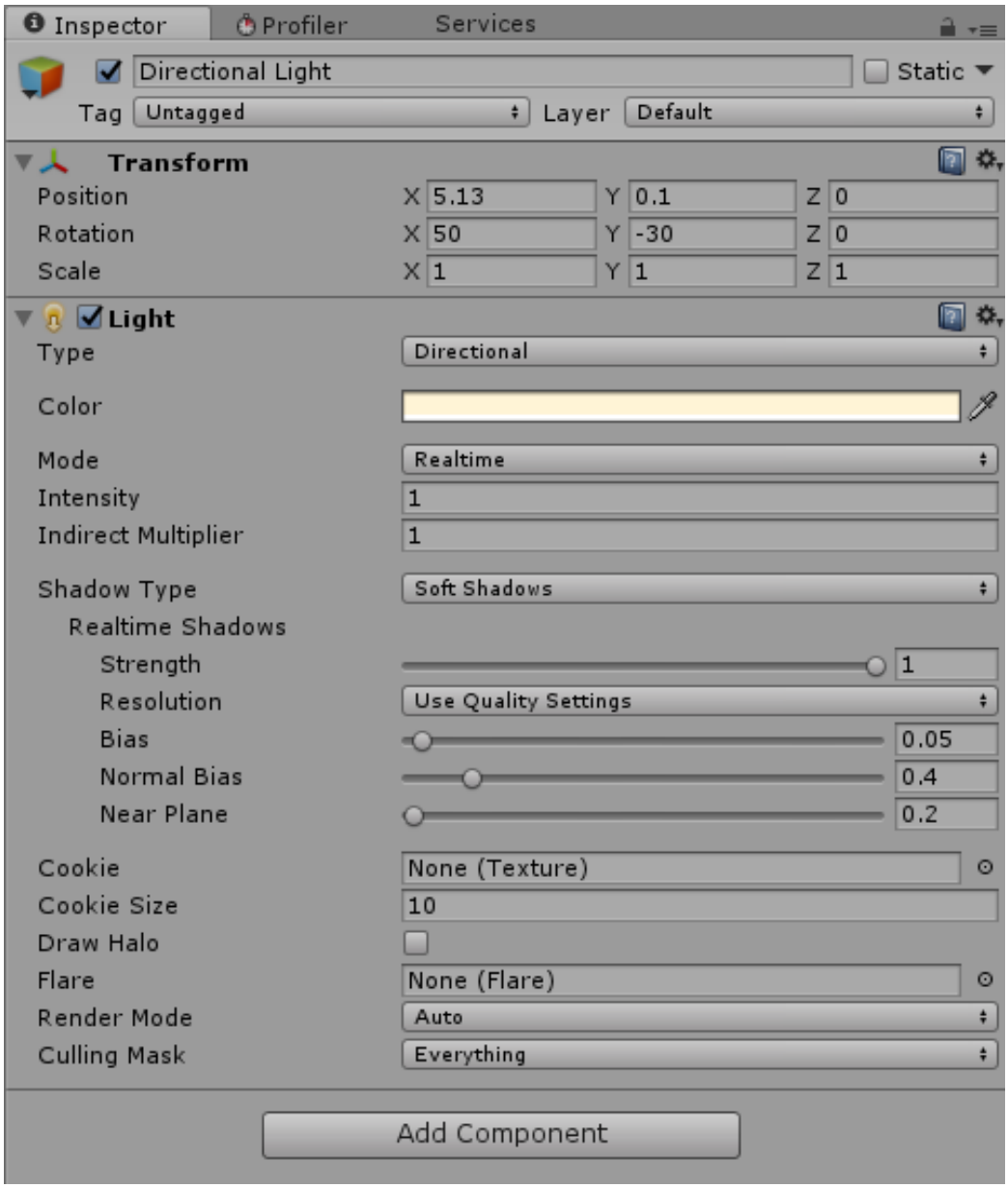
＜Unity 制御画面＞

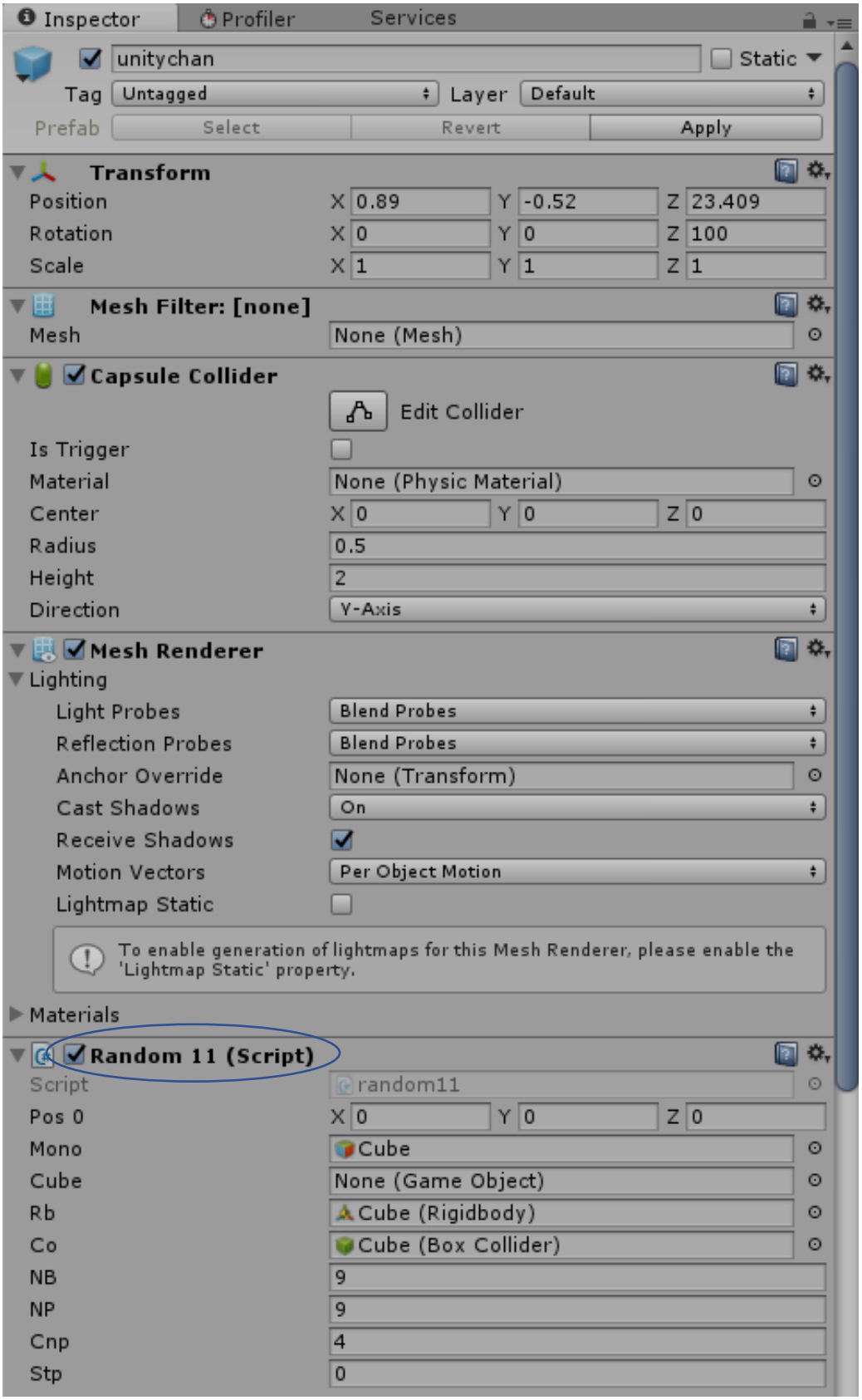


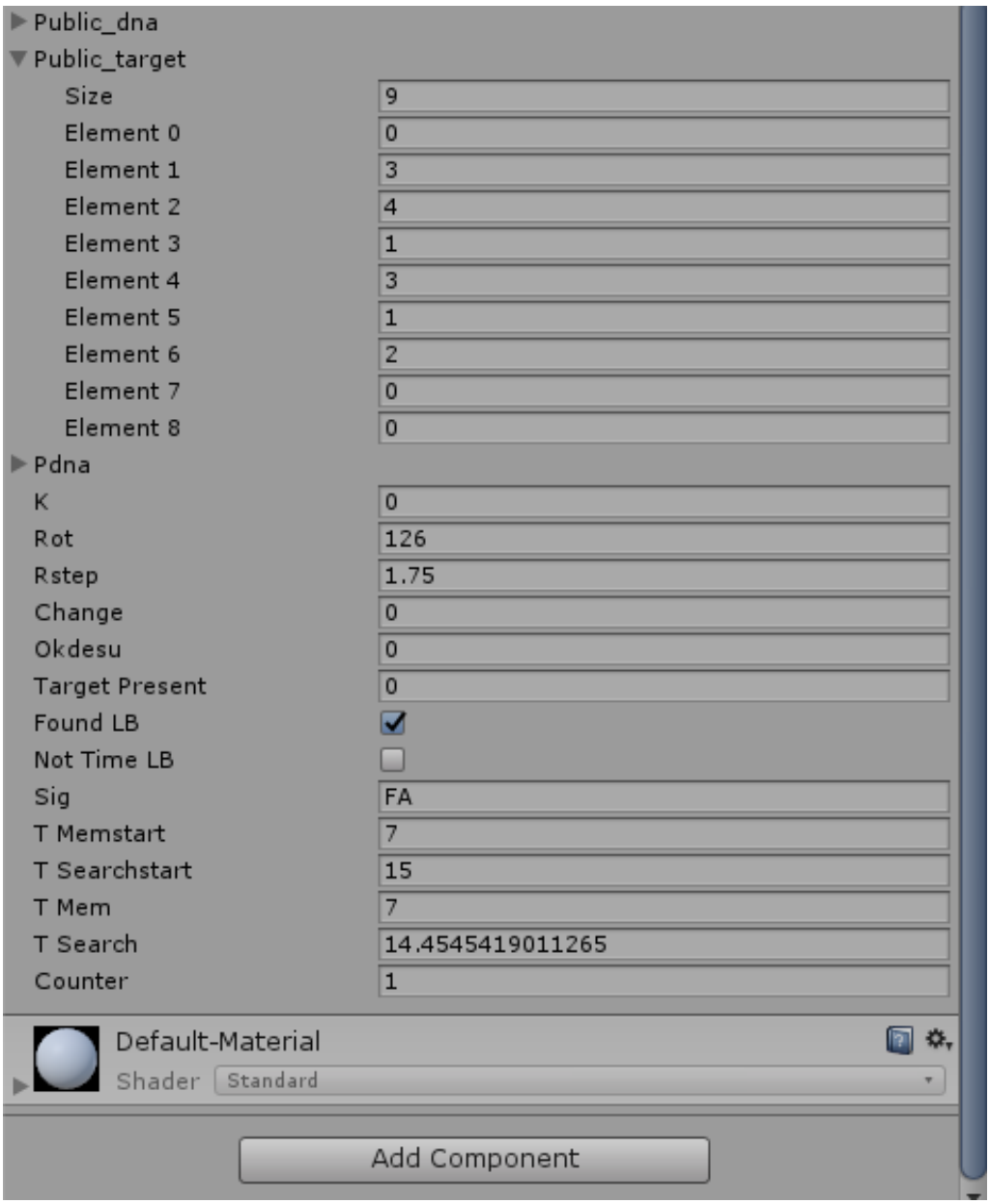
＜インスペクタービュー＞

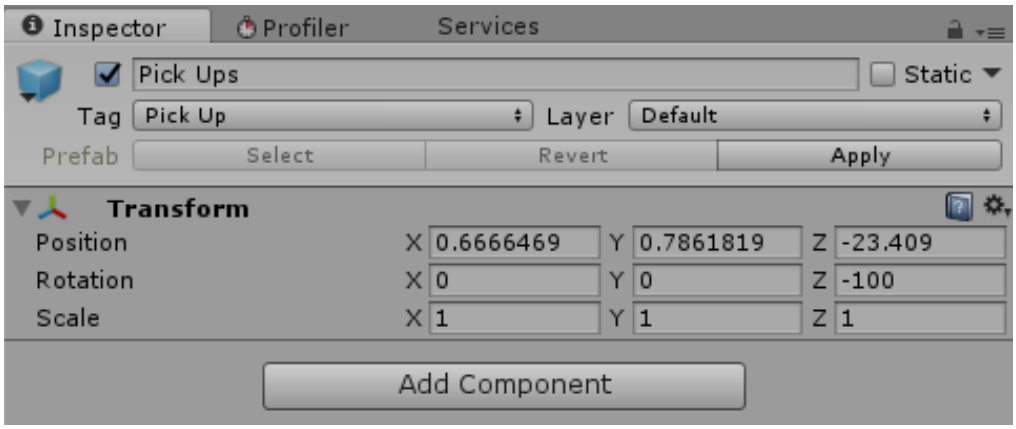
・・・各ゲームオブジェクトの属性（特徴）を表示する。位置、角度、大きさ、どんなプログラムコードで動くか等々、ゲームオブジェクトの設定をまとめて表示する役割がある。

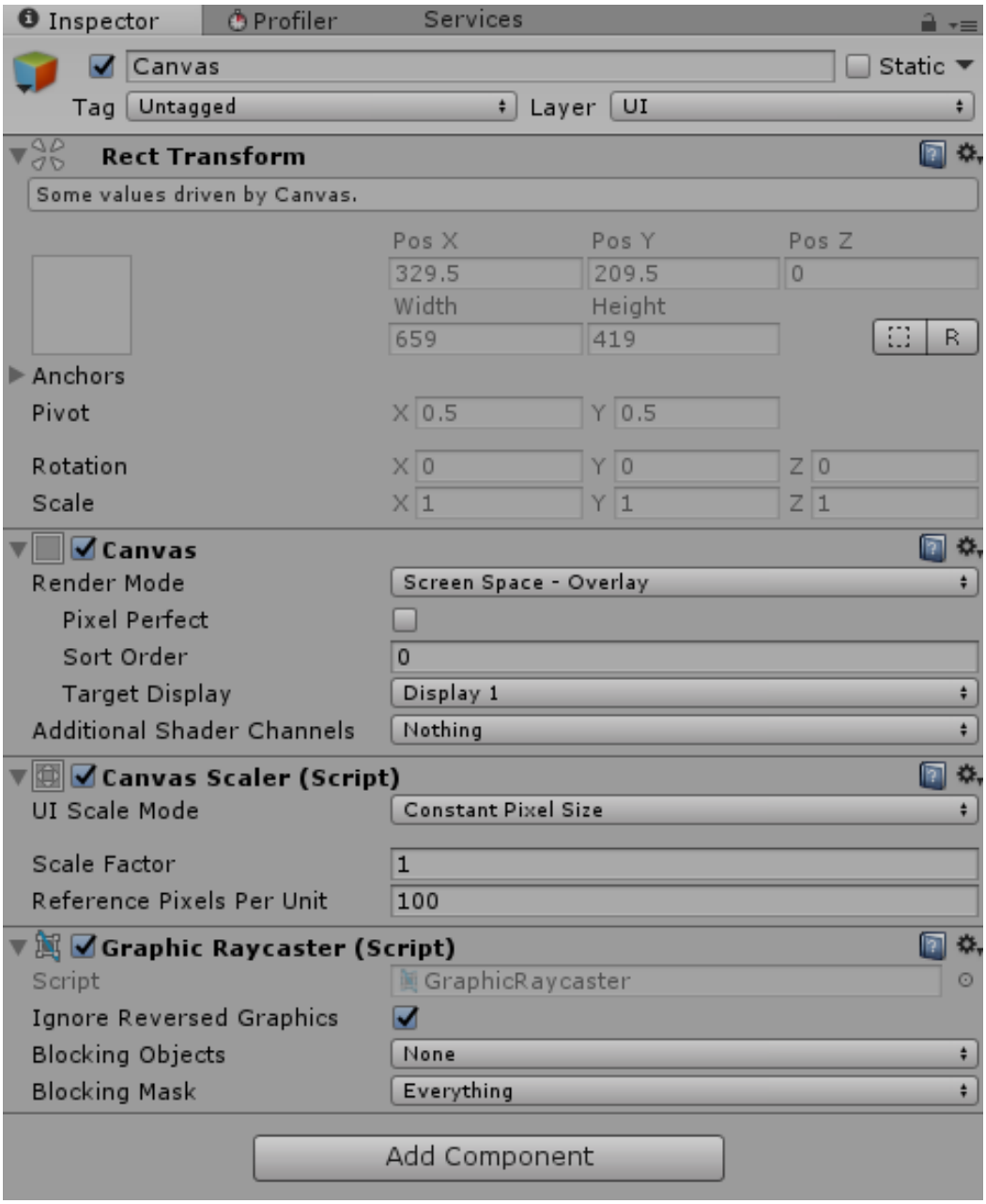
次ページ以降、初期状態の Hierarchy に存在するオブジェクトそれぞれのインスペクタービューを示す。

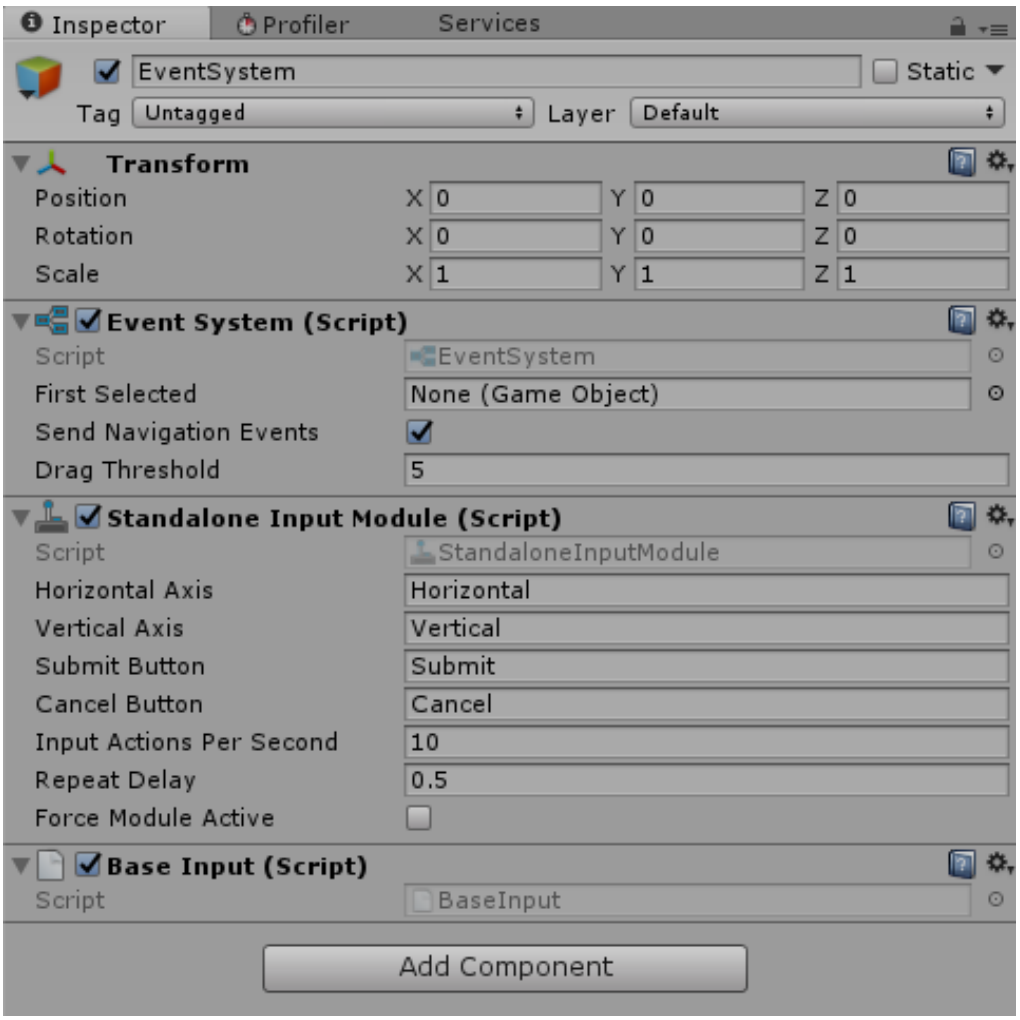

Inspector
Profiler
Services
Directional Light
Static
Tag
Untagged
Layer
Default
Transform
Position
X 5.13
Y 0.1
Z 0
Rotation
X 50
Y -30
Z 0
Scale
X 1
Y 1
Z 1
Light
Type
Directional
Color
Mode
Realtime
Intensity
1
Indirect Multiplier
1
Shadow Type
Soft Shadows
Realtime Shadows
Strength
1
Resolution
Use Quality Settings
Bias
0.05
Normal Bias
0.4
Near Plane
0.2
Cookie
None (Texture)
Cookie Size
10
Draw Halo
Flare
None (Flare)
Render Mode
Auto
Culling Mask
Everything
Add Component

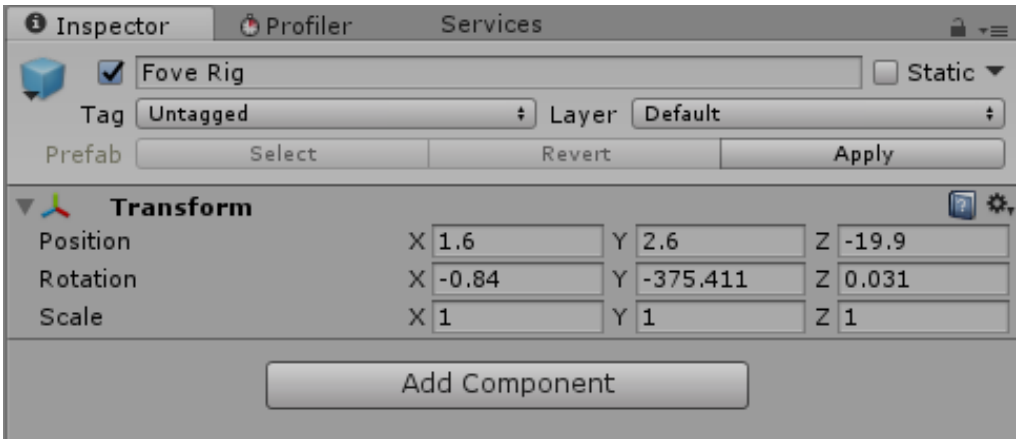

Inspector
Profiler
Services
unitychan
Static
Tag
Untagged
Layer
Default
Prefab
Select
Revert
Apply
Transform
Position
X 0.89
Y -0.52
Z 23.409
Rotation
X 0
Y 0
Z 100
Scale
X 1
Y 1
Z 1
Mesh Filter: [none]
Mesh
None (Mesh)
Capsule Collider
Edit Collider
Is Trigger
Material
None (Physic Material)
Center
X 0
Y 0
Z 0
Radius
0.5
Height
2
Direction
Y-Axis
Mesh Renderer
Lighting
Light Probes
Blend Probes
Reflection Probes
Blend Probes
Anchor Override
None (Transform)
Cast Shadows
On
Receive Shadows
Motion Vectors
Per Object Motion
Lightmap Static
To enable generation of lightmaps for this Mesh Renderer, please enable the 'Lightmap Static' property.
Materials
Random 11 (Script)
Script
random11
Pos 0
X 0
Y 0
Z 0
Mono
Cube
Cube
None (Game Object)
Rb
Cube (Rigidbody)
Co
Cube (Box Collider)
NB
9
NP
9
Cnp
4
Stp
0

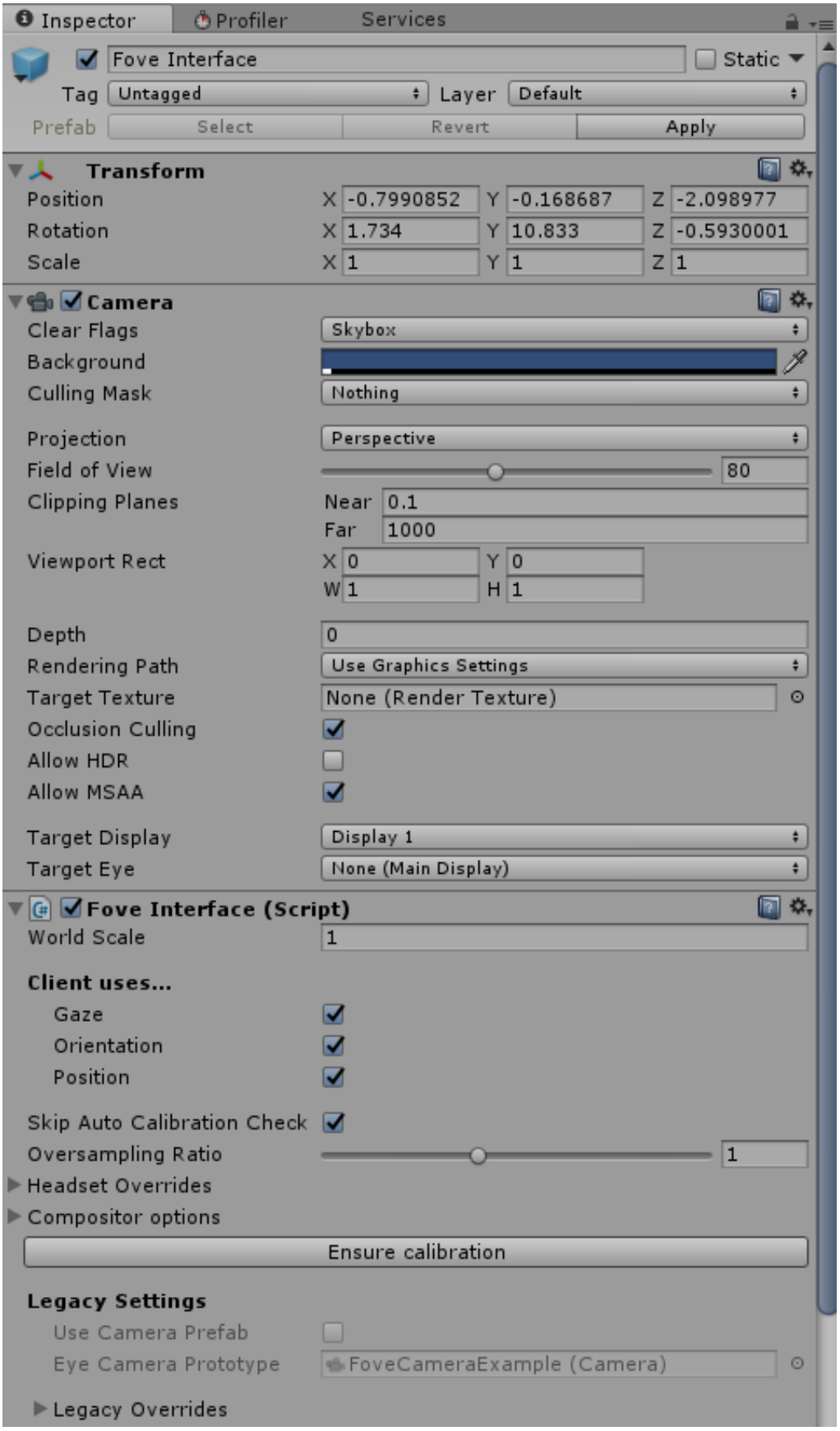

Public_dna
Public_target
Size 9
Element 0 0
Element 1 3
Element 2 4
Element 3 1
Element 4 3
Element 5 1
Element 6 2
Element 7 0
Element 8 0
Pdna
K 0
Rot 126
Rstep 1.75
Change 0
Okdesu 0
Target Present 0
Found LB
Not Time LB
Sig FA
T Memstart 7
T Searchstart 15
T Mem 7
T Search 14.4545419011265
Counter 1
Default-Material
Shader Standard
Add Component

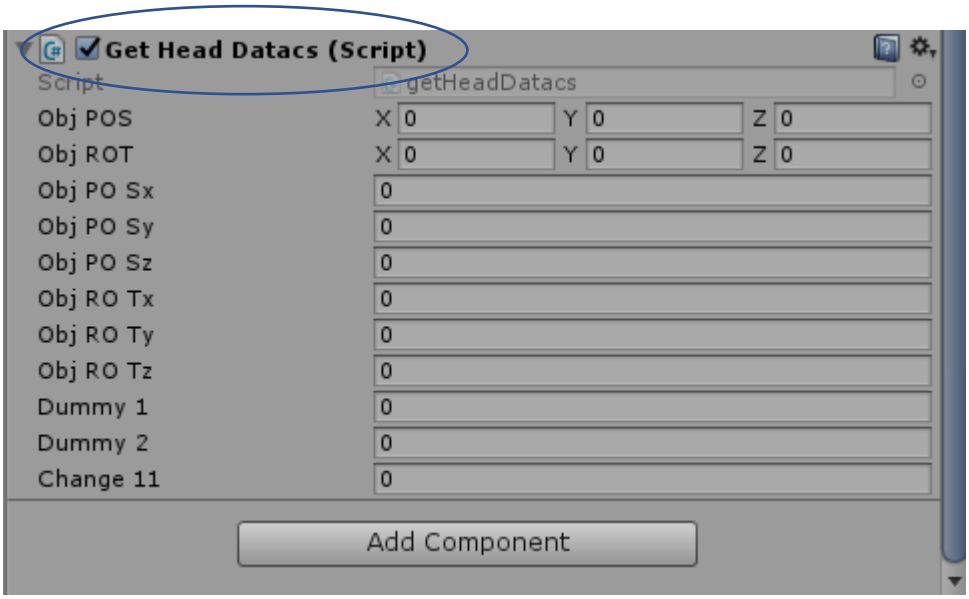

Inspector
Profiler
Services
Pick Ups
Static
Tag Pick Up
Layer Default
Prefab
Select
Revert
Apply
Transform
Position X 0.6666469 Y 0.7861819 Z -23.409
Rotation X 0 Y 0 Z -100
Scale X 1 Y 1 Z 1
Add Component


Inspector
Profiler
Services
Canvas
Static
Tag
Untagged
Layer
UI
Rect Transform
Some values driven by Canvas.
Pos X
Pos Y
Pos Z
329.5
209.5
0
Width
Height
659
419
R
Anchors
Pivot
X 0.5
Y 0.5
Rotation
X 0
Y 0
Z 0
Scale
X 1
Y 1
Z 1
Canvas
Render Mode
Screen Space - Overlay
Pixel Perfect
Sort Order
0
Target Display
Display 1
Additional Shader Channels
Nothing
Canvas Scaler (Script)
UI Scale Mode
Constant Pixel Size
Scale Factor
1
Reference Pixels Per Unit
100
Graphic Raycaster (Script)
Script
GraphicRaycaster
Ignore Reversed Graphics
Blocking Objects
None
Blocking Mask
Everything
Add Component


Inspector
Profiler
Services
EventSystem
Static
Tag
Untagged
Layer
Default
Transform
Position
X 0
Y 0
Z 0
Rotation
X 0
Y 0
Z 0
Scale
X 1
Y 1
Z 1
Event System (Script)
Script
EventSystem
First Selected
None (Game Object)
Send Navigation Events
Drag Threshold
5
Standalone Input Module (Script)
Script
StandaloneInputModule
Horizontal Axis
Horizontal
Vertical Axis
Vertical
Submit Button
Submit
Cancel Button
Cancel
Input Actions Per Second
10
Repeat Delay
0.5
Force Module Active
Base Input (Script)
Script
BaseInput
Add Component


Inspector
Profiler
Services
Fove Rig
Static
Tag
Untagged
Layer
Default
Prefab
Select
Revert
Apply
Transform
Position
X 1.6
Y 2.6
Z -19.9
Rotation
X -0.84
Y -375.411
Z 0.031
Scale
X 1
Y 1
Z 1
Add Component


Inspector
Profiler
Services
Fove Interface
Static
Tag
Untagged
Layer
Default
Prefab
Select
Revert
Apply
Transform
Position
X -0.7990852
Y -0.168687
Z -2.098977
Rotation
X 1.734
Y 10.833
Z -0.5930001
Scale
X 1
Y 1
Z 1
Camera
Clear Flags
Skybox
Background
Culling Mask
Nothing
Projection
Perspective
Field of View
80
Clipping Planes
Near 0.1
Far 1000
Viewport Rect
X 0
Y 0
W 1
H 1
Depth
0
Rendering Path
Use Graphics Settings
Target Texture
None (Render Texture)
Occlusion Culling
Allow HDR
Allow MSAA
Target Display
Display 1
Target Eye
None (Main Display)
Fove Interface (Script)
World Scale
1
Client uses...
Gaze
Orientation
Position
Skip Auto Calibration Check
Oversampling Ratio
1
Headset Overrides
Compositor options
Ensure calibration
Legacy Settings
Use Camera Prefab
Eye Camera Prototype
FoveCameraExample (Camera)
Legacy Overrides



（インスペクタービュー終わり）

次ページ以降、本実験で使用した C#プログラミングコードを
Random11・Get Head Data の順に示す。

＜C#プログラミングコード＞

<Random11＞

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System.IO;

public class random11: MonoBehaviour {
    public Vector3 pos0; // start position
    public GameObject mono; // the object type
    public GameObject cube; // the object type
    public Rigidbody rb;
    public Collider co;//これがないと存在できない
    public int nB = 9; // number of blocks
    public int nP = 9; // number of panel images
    public int Cnp; // directions of positioning
    public int stp; // steps of positioning
    public int[] public_dna; // the current shape "dna"
    public int[] public_target; // the target "dna"
    public int[,] all_dna; // all the dnas of the panel
    public int[] pdna; // the temporary dna
    static string tagname = "block";
    public int k=0 ; // count
    public double rot=0; // the initial rotation
    public double rstep; // the rotation step in DEG
    public int change=0 ; // pressed or not
    public int okdesu=0; // already shown
    public int TargetPresent=0; // the target exists
    public bool foundLB=false; // target found or not
    public bool notTimeLB=false; // target found or not
    public string sig; // the signal HIT, MISS, ABS or FA
    public double tMemstart =7; // memorize time
    public double tSearchstart =15; // search time
    public double tMem ; // memorize variable
    public double tSearch ; // search variable
    public int counter=0; // count problems
    public const int counterMax = 52;
    string DATAFILE = "correctrate-.csv";

    public void DeleteAll(){
        foreach (GameObject o in GameObject.FindGameObjectsWithTag("block")) {
            Destroy(o);
        }
    }

    public int[] mkdna(int N)
    {
        int[] kdna = new int[N];
        for (int i = 0; i < N; i++)
        {
            Cnp = Random.Range (0, 6);
            kdna [i] = Cnp;
        }// for i
        return kdna;
    }// mkdna

    public int[,] makeManydna()
    {
        int xmn = -10; // x min
        int xmx = 10; // x max
        int stpx = 10; // the stepx between objects
        int stpy =10; // the stepx between objects
```

```
        int ymn = -10; // y min
        int ymx = 10; // y max
        int [,] adna = new int[nP,nB];
        pos0= new Vector3(0,0,0);
        TargetPresent = 0;
        for (int ii=0; ii<nP; ii=ii+1)
        {
            float X = Random.value;
            float p = 1f / 18.0f; // the probability to find the target
            if (X < p && TargetPresent==0) {
                TargetPresent = 1;
                for (int jj=0; jj<nB; jj=jj+1){adna[ii,jj]=public_target[jj];}; // assign the dna
            }
            else
            {       int[] tdna=mkdna(nB);
                for (int jj=0; jj<nB; jj=jj+1){adna [ii, jj] = tdna [jj];}; // assign the dna
            }//else
        }  //for ii
        return adna;
    }

    public void makeManyCubes(int [,] adna, double rrot)
    {
        int xmn = -10; // x min
        int xmx = 10; // x max
        int stpx = 10; // the stepx between objects
        int stpy =10; // the stepx between objects
        int ymn = -10; // y min
        int ymx = 10; // y max
        int kk=0; // count images
        pdna = new int[nB]; // the temporary dna
        pos0= new Vector3(0,0,0);
        for (int x=xmn; x<=xmx; x=x+stpx)
        {
            for (int y=ymn; y<=ymx; y=y+stpy)
            {
                Vector3 pos=new Vector3(pos0.x+x,pos0.y+y,pos0.z);
                for (int jj = 0; jj < nB; jj = jj + 1)
                {
                    pdna[jj] = adna [kk,jj];
                }
                mkShape (pdna, PrimitiveType.Cube, pos, rrot);
                kk = kk + 1; }  //for y
        }  //for x
    } // makemanycubes

    public void mkShape(int [] dna, PrimitiveType M, Vector3 pos, double rotY)
    { Vector3 pp = pos;
        for (int ob=0;ob<dna.Length;ob++) {
            mono = GameObject.CreatePrimitive (M);//キューブを創る
            mono.tag = "block";
            mono.AddComponent<Rigidbody>();//キューブにリジットボディーという要素を追加
            rb = mono.GetComponent<Rigidbody>();//rbはキューブにリジットボディーの要素を与えるというのを定義
            co = mono.GetComponent<Collider>();//キューブにColliderの要素を加える
            rb.useGravity = false;//重力がかからないようにする
            rb.isKinematic = false;//静止しないようにする
            co.isTrigger = true;//衝突しても無視する
            if (dna[ob] == 0) {pos.x = pos.x + 1;}
            if (dna[ob] == 1) {pos.x = pos.x - 1;}
            if (dna[ob] == 2) {pos.y = pos.y + 1;}
```

```
            if (dna[ob] == 3) {pos.y = pos.y - 1;}
            if (dna[ob] == 4) {pos.z = pos.z + 1;}
            if (dna[ob] == 5) {pos.z = pos.z - 1;}
            mono.transform.position=pos;
            Vector3 ax = new Vector3 (0, 1, 0);
            mono.transform.RotateAround (pp, ax, (float)rotY);
        }// for ob
    }// mkShape

    void makeTarget(int[] dna, double rrot)
    {
        posO= new Vector3(0,0,0);
        mkShape(dna, PrimitiveType.Cube, posO, rrot);
    }

    void Start () {
        tMem = tMemstart;

        tSearch = tSearchstart;
        rstep=1.75; // the rotation step in DEG
        all_dna = new int[nP,nB]; // the dna initialization
        string dateTime = System.DateTime.Now.ToString ();
        StreamWriter sw = File.AppendText(DATAFILE);
        sw.WriteLine (" **** New Trial: "+dateTime+" ****");
        sw.Close ();
    }

    public void writedata()
    {
        StreamWriter sw = File.AppendText(DATAFILE);
        int HITLB=0;
        if (notTimeLB==true) {HITLB = -2;sig= "NOT";notTimeLB = false;}
        if (TargetPresent==1&&foundLB==true&&notTimeLB==false) {HITLB = 1;sig= "HIT";}
        if (TargetPresent==1&&foundLB==false&&notTimeLB==false) {HITLB = -1;sig= "MISS";}
        if (TargetPresent==0&&foundLB==false&&notTimeLB==false) {HITLB = 0;sig = "ABS";}
        if (TargetPresent==0&&foundLB==true&&notTimeLB==false) {HITLB = 9;sig= "FA";}
        sw.WriteLine (Time.time+", "+TargetPresent+", "+foundLB+", "+HITLB+", "+sig);
        sw.Close ();
    }// writedata

    public void writememtime()
    {
        StreamWriter sw = File.AppendText(DATAFILE);
        sw.WriteLine (Time.time+", MEMORIZED");
        sw.Close ();
    }// writememtime
    public void writesearchtime()
    {
        StreamWriter sw = File.AppendText(DATAFILE);
        sw.WriteLine (Time.time+", SEARCHED");
        sw.Close ();
    }// writememtime

    public void waitKey()
    {
        if (change == 0) {
            if (tSearch <= 0) {
                writesearchtime ();
                change = 1;
                tSearch = tSearchstart; // reset clock
```

```
                notTimeLB=true;
                writedata();
            }
            if (Input.GetKeyDown (KeyCode.LeftControl)) {
                tSearch = tSearchstart;
                tMem = tMemstart; // reset mem clock
                foundLB = true;
                change = 1; // change 0
                writedata();
            }
            if (Input.GetKeyDown (KeyCode.RightControl)) {
                tSearch = tSearchstart;
                tMem = tMemstart; // reset mem clock
                foundLB = false;
                change = 1; // change 0
                writedata();
            }
        } // if change 0
        if (change==1){
            if (tMem <= 0) {
                Debug.Log ("end time");
                writememtime ();
                change = 0;
                tMem = tMemstart;
            }
        } //if change 1
    }// waitkey

    void Update()
    {
        rot = rot + rstep; // rotation
        if (rot>=360) {rot=0;} // reset rotation angle
        waitKey ();
        if (change == 1) {
            tMem = tMem - Time.deltaTime; // count time for memorize
            if (okdesu == 0) {
                DeleteAll ();
                public_target=mkdna (nB);
                makeTarget (public_target, rot);
                okdesu = 1;
            }// if ok desu
            if (okdesu == 1)
            {
                DeleteAll ();
                makeTarget (public_target, rot);
            }
        }//if
        if (change == 0)
        {
            tSearch = tSearch - Time.deltaTime; // count time for search
            if (okdesu == 1) {
                DeleteAll ();
                all_dna=makeManydna ();
                makeManyCubes (all_dna, rot); //(double)0);
                counter=counter+1; // count !
                okdesu = 0;
            }
            if (okdesu==0)
            {DeleteAll ();
                makeManyCubes (all_dna, rot); //(double)0);
```

```
            } // ok desu
        }//if change
        if (counter >= counterMax)
        {
            UnityEditor.EditorApplication.isPlaying = false;
        }
    }// update
} // program end
```

＜Get　Head　Data＞

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System;
using System.IO;

public class getHeadDatacs : MonoBehaviour {

    string FILE_NAME = "motion-.csv";
    int time = 0;
    int interval = 1;
    int i; //counter for beeps
    public Vector3 objPOS;
    public Vector3 objROT;
    public float objPOSx;
    public float objPOSy;
    public float objPOSz;
    public float objROTx;
    public float objROTy;
    public float objROTz;
    public float dummy1;
    public float dummy2;
    public int change11;
    private int counter11;

    FoveInterfaceBase fove;
    Ray right;
    Ray left;

    // Use this for initialization
    void Start () {
        fove = this.GetComponent<FoveInterfaceBase> ();
        Debug.Log ("Start Camera script");
        string dateTime = System.DateTime.Now.ToString ();
        StreamWriter sw = File.AppendText(FILE_NAME);
        sw.WriteLine (" **** New Trial: "+dateTime+" ****");
        sw.Close ();
    }
    float timeLeft=10; // forty secs

    // Use this for repeat
    void Update () {
        GameObject unitychan = GameObject.Find ("unitychan");
        random15 instance11 = unitychan.GetComponent<random15> ();
        change11 = instance11.change; // the variable change from random1111
        counter11 = instance11.counter; // the counter from random11
        right = fove.GetGazeRays ().right;
        left = fove.GetGazeRays ().left;

        if (Input.GetKeyDown("left shift"))
        {
            UnityEditor.EditorApplication.isPlaying = false;
        }

```

```
55          if (time > interval)
56          {// "this" の意味はこのもの！
57              float objPOSx = this.transform.position.x;
58              float objPOSy = this.transform.position.y;
59              float objPOSz = this.transform.position.z;
60              float objROTx = this.transform.eulerAngles.x;
61              float objROTy = this.transform.eulerAngles.y;
62              float objROTz = this.transform.eulerAngles.z;
63
64              StreamWriter sw = File.AppendText(FILE_NAME);
65
66              sw.WriteLine (Time.time + "," + objROTx + "," + objROTy + "," + objROTz+
67
68                  "," + right.origin.x +"," + right.origin.y + "," + right.origin.z +
69
70                  "," + left.origin.x + "," + left.origin.y+"," +left.origin.z+ "," +
71
72                  right.direction.x+"," +right.direction.y+"," +right.direction.z+"," +
73
74                  left.direction.x+"," +left.direction.y+ "," +left.direction.z+"," +
75
76                  counter11+","+change11);
77
78              sw.Close ();
79              time = 0;
80          }
81          else
82              time ++;
83      }
84  }
```