公立はこだて未来大学 2022 年度 システム情報科学実習
グループ報告書

Future University-Hakodate 2022 System Information Science Practice

Group Report

プロジェクト名

自分の気持ちと世界をつなぐ X-Reality とスマートウォッチ

Project Name

X-Reality and Smartwatch Connecting Your Feelings and the World

グループ名

スマートウォッチグループ/AR グループ

Group Name

Smartwatch Group/AR Group

プロジェクト番号/Project No.

3

プロジェクトリーダ/Project Leader

丹野陽翔　Haruto Tanno

グループリーダ/Group Leader

熊坂 賢人　Kento Kumasaka
網干 界　Kai Aboshi

グループメンバ/Group Member

八木田光　Hikaru Yagita
安澤龍生　Ryuki Yasuzawa
尾崎篤史　Atsushi Ozaki
小田涼平　Ryouhei Oda
宮崎隼　Hayato Miyazaki
川上龍仁　Ryuuto Kawakami
森皐太　Kouta Mori

指導教員

佐藤仁樹教授　新美礼彦教授

Advisor

Prof. Hideki Satoh　Prof. Ayahiko Niimi

提出日

2023 年 1 月 27 日

Date of Submission

January 27, 2023

概要

AR(Augmented Reality) 技術とスマートウォッチ・活動量計を用いて，自分の気持ちと世界をつなぐための情報リンクの作成を目的とした．まず，ANT+ 規格に対応したスマートウォッチ・活動量計から取得したデータをリアルタイムで可視化した．次に，Unity 技術 (平面検知，タグ検知，オブジェクト生成，当たり判定) を用いて AR ダーツゲームを開発した．さらに，スマートウォッチ・活動量計からリアルタイムでコンピュータ内に取得した心拍数と飛行機の高度が連動するようなゲームを開発した．このゲームでは，ユーザーが指定した目標値心拍数及び心拍数変動幅を調節することで，運動をはじめとした活動を楽しくサポートすることを目的とした．本ゲームでは心拍数をリアルタイムで可視化できる．これにより，ユーザーの心拍数の瞬間値だけでなくその履歴を取り扱えるようになった．また，Unity 技術により飛行機の挙動の再現，背景の変化，およびスコアの加算などの要素を取り入れることができた．このようにして我々の身体データから，スマートウォッチ・活動量計・コンピュータを通して目に見えるものに繋げるという情報リンクを作成した．しかし，本プロジェクト内においては実験や発表会で 10 回ほど利用したが，ユーザーの興味・関心を引き出すことができたかを客観的に評価できなかった．

キーワード　AR, Unity, スマートウォッチ, 心拍数, ANT+, ゲーム

（※文責: 八木田光）

Abstract

The purpose of this project was to make an information link between one's feelings and the world using AR (Augmented Reality) technology and a smartwatch/activity meter. First, data obtained from smartwatches/activity meters that support the ANT+ standard were visualized in real time. Next, we developed an AR darts game using Unity technology (plane detection, tag detection, object generation, and hit detection). Furthermore, we developed a game in which the plane's motion is linked to the heart rate obtained in real time from a smartwatch/activities meter in the computer. The purpose of this game was to support fun activities such as exercise by adjusting the user-specified target heart rate and heart rate variability range. The game visualizes the heart rate in real time. This makes it possible to handle not only the instantaneous value of the user's heart rate but also its history. Unity technology was also used to show airplane, the background, and scores on the game. In this way, we made an information link from our body data to something tangible through smartwatches/activity meters, and computers. However, in this project, we used it about 10 times in experiments and presentations, but we could not objectively evaluate whether it was able to draw out the interest of users.

Keyword AR, Unity, Smartwatch, Heart rate, ANT+

（※文責: 八木田光）

目次

第 1 章　はじめに

この章では，本プロジェクトと関連のある背景や事例について述べる．また，問題点を挙げる．

（※文責: 安澤龍生）

1.1　背景

近年のコロナ禍での情勢を受けて，2022 年に第一生命経済研究所が行った調査によると，「運動不足だと感じている」という質問に対して『あてはまる』（「あてはまる」または「どちらかといえばあてはまる」）と答えた人の割合が 71.5 %となっている．また，運動習慣を見直したかどうかについての質問に対して，見直した人（「あてはまる」または「どちらかといえばあてはまる」）と答えた人は，33.0 %であり，残りの 67.0 %の人は見直せていないことがわかる [1]．このように，コロナ禍における運動不足の問題が指摘されるようになり，家の中で行う運動なども提唱された．しかし，感染拡大から 2 年数カ月の間に運動習慣を見直した人は 3 分の 1 程度であり，新しい運動・スポーツを始めた人は少ないことがわかった．このような背景から，本プロジェクトでは運動をメインテーマとした．また，本プロジェクトはスマートウォッチから取得できるデータを活用することが目的である．その取得・活用データの対象として，運動中に変動が発生する心拍数に着目した．次に，普段運動することが無くなった人にどうしたら運動してもらえるようになるか検討した．その結果，心拍数と運動を基にしたゲームを制作することで，無意識にゲームをログインしているような感覚で運動を始めてもらえるのではないかと考えた．特にゲームと運動の関係性として，人気ゲームを多く販売している任天堂では，運動をテーマにしたゲームソフト「リングフィットアドベンチャー」を発売している [2]．このソフトでは，ゲーム要素としてスクワットなどの運動が取り入れられており，ゲームを遊ぶことで運動を行うことができる．このソフトはコロナ禍以前の 2019 年 11 月の発売ではあるが，コロナ禍になるにつれて爆発的にヒットすることとなった．このように，ゲームという媒体から運動を位置づけることで，運動を行わなくなった人でもゲームが運動をするきっかけになると考える．また，心拍数を普段から意識するという習慣をつけることで，自身の運動強度を求めることができ，ダイエットやトレーニングなどを行う上で自身の指標を見つけ出すこともできるといわれている．このことから，ユーザーに心拍数を意識させるゲームは有用だと考える．

（※文責: 安澤龍生）

1.2　目的

1.1 の背景でも挙げたように，運動習慣を見直した人は少なく，新しい運動・スポーツを始めた人も少ないことから，それらの人でも気軽に運動を始められ，運動を継続してもらえるようにするためにゲーム開発を行う．よって，本プロジェクトは Unity を用いてゲーム開発を行い，心拍数を意識しながら運動を行ってもらい，運動を新たな習慣の一部になってもらうことを目的とする．そこで本プロジェクトのプロジェクトメンバーの運動データ（ここではランニングマシンを利用

する）を基にゲーム開発に取り掛かる．前期では，スマートウォッチグループと AR グループの 2 つのグループに分けた（以下スマートウォッチグループを SWG，AR グループを ARG とする）．SWG は，運動（心拍数）側からデータの収集や技術の習得に取り組み，ARG は，主に Unity の技術習得をメインに AR ゲームの開発に取り組む．後期からは 2 つのグループが前期の成果をお互いに共有し，また中間発表会で得た情報などを基に，そこで起こる問題点や不足分などをあぶりだし，最終成果物としてゲームの開発に取り組む．

本プロジェクトは，今年度から始まったプロジェクトのため引き継ぎ作業などはなかったが，次年度に引き継いでもらうためにも，面白くやりがいのありわかりやすい成果物にする必要がある．そのため，個々では慣れない作業もあるが次年度にも引き継げるようにまとめながら作業していく．

（※文責: 安澤龍生）

1.3 従来例

1.1 でも述べたように運動をゲームに組み込んだ例として，任天堂のゲームソフト「リングフィットアドベンチャー」がある．このソフトは付属のリングコンという専用コントローラーを持ちながら運動を行うことがメイン機能となっている．そして，筋トレ系・リズム系・ヨガ系などのジャンルに分かれた運動メニューを行い，腕・お腹・足などの部位を鍛えることが目的となっている．さらに，運動後はコントローラーのセンサーを利用して心拍数を計測することができる．また，他にゲームと心拍数の関係性を探してみると，心拍数を使ったゲームは見つけることができなかった．しかし，1 つのエンターテインメントとして，YouTube などの動画投稿サイトでは，ホラーゲームなどの心拍数の増減が激しいゲームを行う上で心拍数を動画上に可視化している動画が近年増加している．そのため，心拍数とゲームの関係性というものは近年，重宝されてきていると言っても過言ではない．また，ランニングマシン中はただ単調になるため，ジムなどではテレビを設置したり，家でも動画をみながらランニングマシンを使うことで飽きずに運動を行うようにする「ながら運動」が多く存在する．以前，函館アリーナのトレーニングルームを利用した際，ランニングマシンのところにテレビがあり，それを見ながらランニングを行うことで楽しみながらトレーニングをすることができるようになっていた．このように「ながら運動」は，運動を始めるためには必須になってくるのではないかと思っている．

これらは，心拍数のデータを基にしたゲームではなく，心拍数を 1 つの要素とする例であった．デジタルアーティストのエリン・レイノルズさんが開発中の「Nevermind」では，人間の心拍数を計測して，そのデータによってゲームの内容が変化するというものとなっている [3]. このゲームによって，ホラー要素が苦手な人には内容が易しくなったり，反対にホラー要素が得意な人にはよりインパクトの強い内容にすることで，1 つのゲームで多様的にユーザーを獲得できる．

しかし，これらの例も本プロジェクトで行いたいこととは違う目的であるなど，本プロジェクトの従来例としては問題点が存在するため，1.4 では，従来例ごとに存在する問題点について述べていく．

（※文責: 安澤龍生）

1.4　従来の問題点

1.3 で挙げた 4 点の問題点として，1 つ目のリングフィットの例では，心拍数を計測することはしているようだが，それを基にしたゲームというものはなく心拍数を意識しながら運動（ゲーム）をするという点とは違う心拍数との向き合い方をしている．そのため，ゲームと運動という点においては良い例になるが，心拍数とゲームの結びつきという点においては違うものとなっている．2 つ目のエンターテインメントとしての心拍数の例では，心拍数を表示しているため，そのデータを可視化させるという点においては良い例になるのだが，YouTube などの動画投稿者はそれ以上に必要としているわけではない．そのため，発展性としては，演出の発展としてはあるが，中身の発展性は見つけることができない．3 つ目の「ながら運動」の例では，主にジムなどでは大きな音を出すことができないため，テレビに関しても音を出さず，字幕での視聴となる．または，事前にイヤホンなどを用意する必要がある．最後に 4 つ目の例は，現在開発中のゲームのため問題点として挙げることは難しい．そのため，現時点では，あまりあげられる問題点はない．しかし，本プロジェクトではこれらの問題点を解決できるように作業を行ってきた．

（※文責: 安澤龍生）

1.5　課題

この節では，プロジェクトを進めていくうえで設定した課題の概要について述べる．

（※文責: 安澤龍生）

1.5.1　新規プロジェクト

はじめの課題として新規のプロジェクトであることが挙げられる．他のプロジェクトの中には昨年以前から続いているプロジェクトも存在し，学部 4 年生以上の先輩学生がそのプロジェクトの補助をするということも可能だったであろう．しかし，本プロジェクトはそのような先輩学生は存在せず，新しくスマートウォッチを利用するということから他のプロジェクトとはスタートラインの違いが存在する．そのため，どのようにプロジェクトを進めていくのかを決める必要がある．さらに来期以降引き継いでもらうためにプロジェクト全体を整理することも必要となる．

（※文責: 安澤龍生）

1.5.2　スマートウォッチ・活動量計・その他器具の選定

2 つ目の課題は，スマートウォッチ・活動量計・その他器具の選定の選定である．スマートウォッチに絞ってみても，計測できる項目や精度，そのデータを PC に転送できるのかなど多くの違いが存在するため，本プロジェクトの目標に合ったスマートウォッチ・活動量計などを選定する必要がある．それを用いてデータ収集を行うことや，そのゲームの要素として利用する目的がある．

（※文責: 安澤龍生）

1.5.3 リアルタイムでの身体データを PC 上で取得

3 つ目の課題は，リアルタイムで取得した身体データを直接 PC 上で取得することである．1.5.2 で選定したスマートウォッチ・活動量計などのデータを PC にリアルタイムで送信させることでゲームとして活用できると考える．そのため，リアルタイムでのデータの送受信に最も適している ANT+ 規格を利用する．スマートウォッチの代表とも言える Applewacth はデータを送信することができるが，iPhone のアプリのヘルスケアから送信させる手順があり，かつ ANT+ 規格に対応していないため，自動でリアルタイムにデータを送信することは不可能であった．

（※文責: 安澤龍生）

1.5.4 Unity を用いた AR ゲーム制作

4 つ目の課題は，Unity を使用することでゲーム開発に関して学ぶことである．前期の ARG は後期でもゲーム開発班として作業を行っていくために前期ではゲーム開発に向けて学習する必要がある．そのために AR という技術を利用して AR ゲームの開発に取り組んだ．

（※文責: 安澤龍生）

1.5.5 Unity とスマートウォッチの接続

5 つ目の課題は，1.5.3 で取得したデータを Unity を用いてゲームを制作するため，そのデータを Unity へ送信させることである．この課題を解決することで 1.5.7 につながり，ゲーム開発の基盤となるため，作業としても早めに課題解決に取り組む．

（※文責: 安澤龍生）

1.5.6 飛行機などのデザイン

6 つ目の課題はデザインについてである．本プロジェクトメンバーで絵を描くことが得意な学生はいなかったため，絵の描きやすい iPad を所持している安澤が担当した．また，ただ絵を描くだけでは単調なゲームになってしまうため，ここはこだわって作業をしなくてはならない．そのため，多くのメンバー間でアイデアを出し合いながらそれに基づいてデザインを考える．

（※文責: 安澤龍生）

1.5.7 ゲーム開発

最後の課題はゲーム開発である．前期では，Unity を用いた開発を学び，後期からはその学んだ技術を基にゲーム開発を進めていく．1.6.6 と並行作業で行うことで時間の効率と担当者の作業を均等に割り振る．

（※文責: 安澤龍生）

1.6　課題解決に向けて

この節では，1.5 で設定した課題の解決プロセスについて述べる．プロセスの詳細などについては，第 2 章以降で述べていく．

（※文責: 安澤龍生）

1.6.1　新規プロジェクト

本プロジェクトは，新規のプロジェクトであるため進め方などは先生方のアドバイスを基に作業していった．1.2 でも述べたように，前期では，スマートウォッチでのデータ収集とその可視化を目的とし，スマートウォッチ側から本プロジェクトに関わっていく SWG と，Unity でゲーム開発を目的とし，Unity 側から本プロジェクトに関わっていく ARG のグループの 2 つに二分化させた．ちなみに ARG は前期の制作物が AR ゲームであったためその名となった．後期からは，その 2 つのグループを 1 つにすることで効率的に作業していく．最後には来季以降このプロジェクトを継続させるために，これまで使ってきた Google ドライブで共有していた資料やデータ等を整理する．それによって来期以降の後輩にも自分たちが行ってきたことを共有することができる．

（※文責: 安澤龍生）

1.6.2　スマートウォッチ・活動量計・その他器具の選定

本プロジェクトはスマートウォッチを利用するためにプロジェクト内で適切であると考えるスマートウォッチの選定を行った．その中で，血圧やそのほか心電図などの多くの計測が可能であるもの，windows へのデータ転送が可能なものを絞った．その結果，どちらも可能な AppleWatch Serise7 に決定した．また，Applewacth では計測できない血圧や体温を計測するために追加で，選定を行った．しかし，リアルタイムでの作業をするうえで，そのヘルスケアから PC へ転送する作業がどうしても自動化できそうになかった．そこで AppleWatch の便利さを諦め，ANT+ 規格のスマートウォッチとそのドングルの選定を行った．ここで前回 AppleWatch1 台のみしか購入しなかった反省を生かして，スマートウォッチとドングルを複数台購入することにした．これらスマートウォッチ等を選定するためにまとめたリストは付録 A の『デバイスを選定した際の表』に記載している．

（※文責: 安澤龍生）

1.6.3　リアルタイムでの身体データを PC 上で取得

詳しくは 2 章以降で述べているが，ANT+ 規格のスマートウォッチと計測器を利用してデータを取得し，そのデータを ANT+ 規格に対応したドングルで PC に受信させて PC にもデータを所得する．そうすることで，AppleWatch ではできなかった Windows へのリアルタイムでのデータ送信を可能にした．この作業を前期に SWG が担当し，前期の成果物として提出した．

（※文責: 安澤龍生）

1.6.4 Unity を用いた AR ゲーム制作

AR ゲームの開発に関しては，3 章で詳しく述べているが，これまで制作してきたアプリケーションや会得した技術の総まとめとして，AR ダーツアプリケーションを制作することとした．具体的には，AR タグを用いたタグ検知，オブジェクト生成，当たり判定といった機能をすべて導入することを条件としたうえでゲームとして成立させるアプリケーションの制作となる．

（※文責: 熊坂賢人）

1.6.5 Unity とスマートウォッチの接続

データの取得には，JavaScript を利用していたが，Unity は，外部から情報を受け取るために C#での制御が必要であるため，JavaScript から C#に情報を変換または送信する仕組みが必要である．詳しい仕組みなどに関しては 4.4 で述べている．これによって 1.6.7 の作業を行う準備ができた．また時間も多くかかることはなかったため，作業に支障が出ることがなく進めることができた．

（※文責: 熊坂賢人）

1.6.6 飛行機などのデザイン

デザインは，iPad で無料アプリケーションである「ibisPaint X」を用いて行った（制作したデザイン画像は付録 E に記載した）.「ibisPaint X」を利用した理由は，手書きでは不安定になってしまう線を補正することができ，フィルターを利用することで色合いの調節などもできるからである．それによって不安だったデザインも最低限の絵にすることはできた．そのほかにも背景やゲームタイトルのデザインなども行った．

制作物として飛行機のデザインのほかにも背景は，飛行機が離陸するための滑走路，フライト中の外の背景をデザインした．そのほかにも利用するまではいかなかったが，ギミックとして，風船・気球・麦わら帽子など用意していた．このギミックのデザインに関しては，次年度以降参考にしていただけたら幸いである．

（※文責: 熊坂賢人）

1.6.7 ゲーム開発

ゲーム開発に関しては 4.4 で詳細は述べているが，主に前期の ARG が担当した．1.6.5 で Unity へ送信したデータをゲームの要素して利用し，ゲーム開発を行った．その仕様書に関しては，付録 H の『HEARTBEAT AIRLINE 仕様書』で記述している．このゲーム開発に関して，

（※文責: 熊坂賢人）

第 2 章　スマートウォッチを使用した身体データの取得

この章では，前期に SWG が活動した内容について述べる．前期の SWG では本プロジェクトの最終目標を達成するべく，SWG の目標を，「スマートウォッチで取得できる身体データを取り込み分析すること」と「取得したデータをリアルタイムで活用すること」の 2 つに設定した．また，運動やマッサージなどで変化する心拍数をスマートウォッチや活動量計で計測する実験及び，考察を行った．更に，アプリケーションを用いて心拍数の可視化を行った．

（※文責: 熊坂賢人）

2.1　目的・目標

本グループは，「X-Reality とスマートウォッチを用いて身の回りの現象に接続し，自分の気持ちと世界をつなぐための情報リンクを作成する」というプロジェクトの最終目標を達成するために，スマートウォッチの技術を利用して身の回りの現象と接続することを目的としたグループである．スマートウォッチとは，心拍数や歩数といったデータを取得することが可能な腕時計型のウェアラブル端末のことである．本グループでは，スマートウォッチを使って取得が可能な身体データを，この最終目標に活用できると考えた．

この最終目標を達成するために本グループでは，以下の 2 つを前期の目標とした．

- スマートウォッチで取得できる身体データを取り込み分析する（2.2 を参照）
- 取得したデータをリアルタイムで活用する（2.3 を参照）

（※文責: 熊坂賢人）

2.2　身体データに関する実験

2.1 で述べた，スマートウォッチや活動量計でどのような身体データが取得できるかの確認と，過去の身体データからその変動や数値を確認するために，実験用の機材を購入するために使用するスマートウォッチの選定から始め，マッサージをした際の心拍数の計測をする実験を行った．本節では，その実験について述べる．

（※文責: 熊坂賢人）

2.2.1　実験内容

本実験では，スマートウォッチを使用して取得することの出来るデータのうち，特に変動しやすい心拍数を活用することとした．また，心拍数の変動を記録するために，運動とマッサージを実験の対象とした．運動に関する実験では大学のトレーニングルームを利用し，SWG メンバーの 2 人

が運動を行い，運動中と運動前後の心拍数の変動を測定した．実験で行った運動は，ランニングマシンを用いて時速 6km のランニングを 5 分間行うものと，腹筋運動 20 回を 3 セット，セット間に 30 秒のインターバル 30 秒取った上で行うもの，そしてベンチプレス 30 秒を 3 セット，セット間に 20 秒のインターバルを取ったものの 3 種類を行った．なお，身体能力の違いから，ベンチプレスの重さはそれぞれ 40kg・20kg とした．次に，マッサージの実験では，マッサージの専門家として函館視力障碍センター教務課の村松芳容様にご協力いただき，実験を行った．この実験では，SWG メンバー 2 人 (以下被験者 A,B とする) がマッサージを受け，マッサージ中とマッサージ前後の心拍数，そしてマッサージ開始時と終了時の血圧を測定し，その変動を測定した．マッサージの内容について，被験者 A は肩周辺のマッサージを 2 分間，上半身全体のマッサージを 10 分間の計 2 回を，被験者 B は，下半身のマッサージを 7 分間を二回受けた．ただし，それぞれのマッサージの強さを強めたものと弱めたものの二回を行った．

（※文責: 熊坂賢人）

2.2.2　機材の購入について

SWG では，スマートウォッチや活動量計といったデバイスを選定したが．ここでは，その際に使用した，実験に関連したデバイスをまとめたリストと，選定方法をまとめる．SWG では活動の初期で，様々なスマートウォッチを 25 個リストアップした．その後それぞれのデバイスに関して，心拍数や血圧，血中酸素濃度といったスマートウォッチで計測することのできる数値や，取得したデータをどのようにしてスマートフォンや Windows PC に送信するのか，そしてそれぞれのデバイスの特徴と値段をまとめた．それぞれの特徴では，各デバイスで報告されている不具合やそのデバイス専用のソフトウェア，計測頻度などわかったことについて簡単にまとめた．それらについてまとめた表については，付録 A にまとめる．SWG では，取得したデータを可視化することを最初の目標としていたため，デバイスの選定の際，最初に Windows へのデータの転送が可能かどうかを重視した．多くのスマートウォッチはデータの転送が可能であり，それぞれどのようにして転送することができるのか，データを CSV 等に起こすことはできるのかを調べた．その結果，データの出力を 1 日に 1 回しかできないものや出力に数日かかるもの，CSV 等に出力ができない物などを除き再検討した結果，AppleWatch Serise7 を購入することにした．

（※文責: 熊坂賢人）

2.2.3　機材の選定とデータ出力

今回心拍数の測定に用いた機材は，「AppleWatch SE」と「AppleWatch Serise7」の 2 種類のスマートウォッチである．この 2 つのスマートウォッチを用いた理由として，測定の精度が高いことと，手元に「AppleWatch SE」がすでにあり，一部の実験にすぐに取り掛かることができ，「AppleWatch Serise7」も在庫があったため，すぐに用意することができたことが挙げられる．また，この 2 種類は iPhone とのペアリングが可能であるため，測定したデータは iPhone 標準アプリである「ヘルスケア」に追加することができる．その後，ヘルスケアの標準機能で CXV ファイルで計測データを出力した後，そのファイルを Windows PC

血圧の測定には，「上腕式血圧計 HEM-7600T-W」を用いた．用いた理由としては，測定の精度が高いこと・安定した測定が可能であることが挙げられる．この血圧計は Bluetooth によってユー

ザーの iPhone と接続することができるため，取得したデータを「ヘルスケア」に追加することができる．

今回は「ヘルスケア」内に記録されているデータを csv ファイルとして書き出してデータ解析を行った．

（※文責: 熊坂賢人）

2.2.4 実験結果

運動による実験のデータは，データが連続して取得できていない箇所があり取得が上手くいっていなかったためデータ解析を行うことができなかった．

マッサージで計測した血圧を次に示す．

1 人目 1 回目：126 → 133

1 人目 2 回目：126 → 132

2 人目 1 回目：132 → 136

2 人目 2 回目：132 → 141

マッサージによる実験で取得した心拍数のグラフ化を以下に示す．

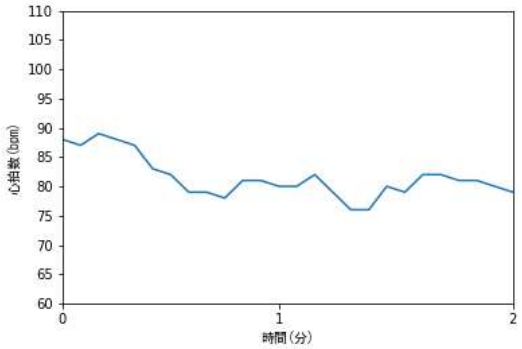

図 2.1　肩回り 2 分間

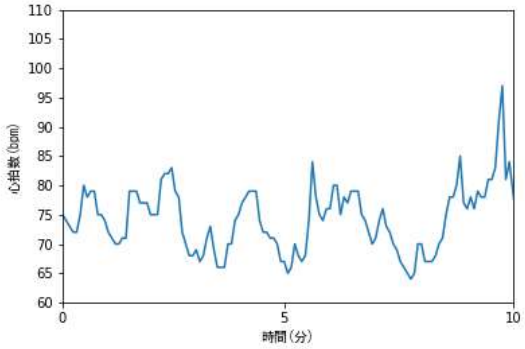

図 2.2　上半身 10 分間

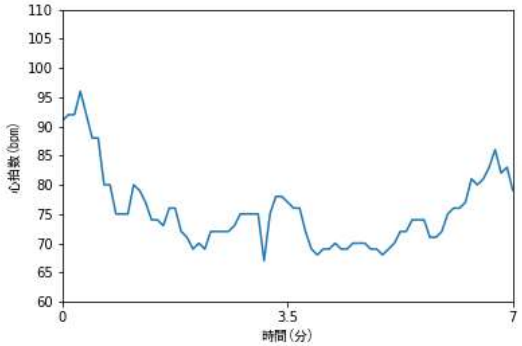

図 2.3　下半身（弱）7 分間

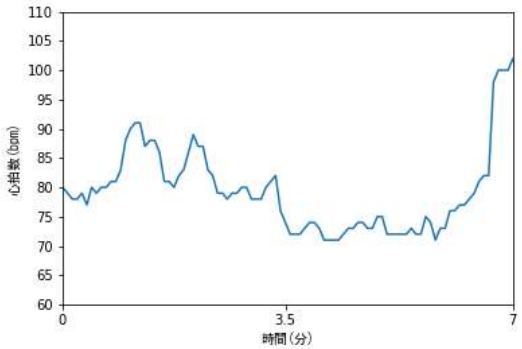

図 2.4　下半身（強）7 分間

（※文責: 熊坂賢人）

2.2.5　考察

一般に，マッサージは心拍数を下げるとされている．例えば若杉ら（2020）はハンドマッサージの受け手の手の温度以上の温かさで実施した場合には，心拍数を低下させ，心理的なリラックス感を提供できることを明らかにした．[4] 今回のマッサージの実験では，肩回りのマッサージは時間経過に伴い心拍数の変化に緩やかな現象が見られた．上半身全体のマッサージの心拍数は細かな上昇・下降をしていたことから，上半身への刺激は心拍数への影響が大きいと考えられる．下半身（弱）のマッサージでは，5 分ごろまでは心拍数の減少が見られたがその後上昇が見られた．これは，下半身の中でもマッサージする場所が変わったからだと考えられる．痛みを伴う下半身のマッサージの心拍数は数回突起した数値があることから，瞬間的に心拍数が上がる場面があったといえる．これらのことから，マッサージを受ける位置や時間によって心拍数が変動すること，痛みは心拍数を上げることが考えられる．血圧はどのマッサージにおいて開始時と比べて終了時が高くなったことから，マッサージをすると血圧が上がる要因があると考えられる．

（※文責: 熊坂賢人）

2.3　可視化アプリケーション概要

2.1 で設定した，取得したデータをリアルタイムで活用するという目標を達成するために，スマートウォッチで取得したデータを Windows の PC にリアルタイムで表示するアプリケーションを利用した．この節では，利用したアプリケーションについての詳細を述べる．

（※文責: 八木田光）

2.3.1　機材の購入について

リアルタイムでデータを活用しようとした際，既存のデバイスでは不可能であり，技術等を調査したところ ANT+ 規格を使用することで可能になることが分かった．そのために，ANT+ を搭載したスマートウォッチや活動量計を購入することにした．一度購入した際のリストを参考にしながら，ANT+ 対応のスマートウォッチを 10 個リストアップした後，その中でも壊れにくく比較的高評価のレビューの多かった vivosmart 4 を購入することにした．それと同時に，ANT+ 対応のドングルも 4 個リストアップした後，評価の良かった Allez と CycleAnt，CooSpo の ANT+ USB ドングルを購入した．

（※文責: 熊坂賢人）

2.3.2　使用した機材

ANT+ 対応のスマートウォッチや活動量計と，専用のドングルを用いて，PC へのデータの送信を行った．ANT+ に関しては 4.4.3 で詳しく記述する．今回スマートウォッチは vivosmart 4，心拍計には COOSPO HW807 と iGP SPORT HR60 を使用した．ANT+ 対応のドングルには，Allez，CycleAnt，CooSpo ANT+ USB ドングルを使用した．また，データの表示には Node.js を用いた．

（※文責: 八木田光）

2.3.3　ソフトの仕様

ANT+ 対応のスマートウォッチや計測器とドングルを用意し，このアプリケーションをダウンロードした PC へドングルを接続する．その後，アプリケーションを起動すると，自動的にスマートウォッチや活動量計と接続がされ，Node.js を通してリアルタイムでグラフに起こした物が html ファイルで表示される．ソフトの使用手順は 2.4 にて述べる．

（※文責: 八木田光）

2.4　リアルタイムでの心拍数変動の確認

ANT+ を利用してリアルタイムで心拍数を可視化させるための手順を説明する．主に次のサイトを参考に作業を行った．[5]

使用するプログラム

- zadig v2.0.1.160.7z (ANT+ 用のドングルを Windows で使用する為のドライバー)
- node-v12.15.0-x64.msi (Node.js のインストーラー)
- HeartBeats-master（心拍数を受信する為のソフトウェア）
- StartHeatBeats.bat（上記のソフトウェア内にある受信できているか確認できるプログラム）
- html/index.html（同じく上記のソフトウェア内にある html でリアルタイムで可視化させたアドレスバー）

注意点

- 心拍計が ANT+ 対応か確認する

必要なもの

- ANT+ に対応した心拍計
- ANT+ に対応したドングル
- Windows PC
- Visual Studio Community 2022（本プロジェクトでは ver 17.4.4 を使用）
- Python 環境（本プロジェクトでは ver 3.11.1 を使用）

セットアップ

1. Visual Studio のホームページ（`https://visualstudio.microsoft.com/ja/vs/whatsnew/`）にアクセスする.
2.「Visural Studio のダウンロード」をクリックし,「Community 2022」を選択.
3. 自動的に「VisualStudioSetup.exe」がダウンロードされるので, そちらを開き「続行」をクリック.
4. ワークロードのタブを開き,「C++ によるデスクトップ開発」と「ユニバーサル Windows プラットフォーム開発」にチェックを入れる.
5. 右側に表示されたインストールの詳細から,「JavaScript 診断」と「Windows 10 SDK(10.0.20348.0)」にチェックを追加する.
6. インストールをクリック.
7. python の公式ページ（`https://www.python.org/`）にアクセスする.
8.「Downloads」から Download for Windows の下にある「Python 3.11.1」をクリック.
9.「python-3.11.1-amd64.exe」を開き,「Install now」をクリックする.
10. ANT+ 用のドングルを Windows で使用するために Zadig というドライバーをインストール.(ここで zadig v2.0.1.160.7z をインストール)
11. 拡張子が 7z のため解凍ソフトなどを利用して解凍する.
12. ドングルを繋ぎ, Zadig を起動させる.
13. ドライバー上記の Options をクリックしその中にある List All Devices にチェックを入れる.
14. 上の枠を ANT USBStick2 を選択.
15. Replace Driver を選択して ANT+ 用のドライバーをインストール.

16. これでドライバーのインストールは終了だが，作業を Node.js で行うため Node.js をインストールする．
17. Node.js の HP よりインストーラーをダウンロード．
18. ダウンロードしたインストーラーを展開し，インストールを行う．
19. Next をクリック，承認のチェックを入れ，Next をクリック，インストール先を選定し，Next をクリック，インストールしたいコンポーネントを選択し，Next をクリック，インストールボタンをクリックし，インストールを開始する．
20. Node.js のインストールは完了したが，確認のためコマンドプロンプトを起動．
21. node –version のコマンドより，Node.js が入っていることとバージョンを確認．

ソフトの起動

1. GitHub から tokjin/HeartBeats をダウンロード（zip ファイルのため解凍も忘れず）
2. コマンドプロンプト or PowerShell を開く
3. cd ***で，先ほど GitHub からダウンロードしてきたフォルダに移動．
4. npm install を実行．
5. 心拍計の電源を入れる．
6. 同封の StartHeatBeats.bat を開く（もしくは node main.js を実行）
7. うまく動いていれば count の数値が常に増えていく．
8. 先ほど GitHub からダウンロードしてきたフォルダの html/index.html を開くと表示される．

ソフト内の説明

- count：心拍計のデータを読み取った数
- beat：実際に計測された心拍数
- id：心拍計の端末に割り当てられた固有 ID
- timeout：count が 10 秒間増えなかった場合タイムアウトしソフトが再起動する．

（※文責: 丹野陽翔）

2.5 中間発表会

ここでは，前期に行った中間発表会（2022,7,8）の内容のうち前期 SWG に関する部分を記述する．

（※文責: 八木田光）

2.5.1 発表準備

本プロジェクトでは前期は SWG と ARG に分かれて活動していたため，各グループが用意した発表内容を順に発表するという形式とした．発表会の形式が前後半（各 3 回）に分かれていたため，SWG 内で前半（尾崎，安澤）と後半（丹野，熊坂，八木田）に分けた．各グループで発表スライドを作成し 1 つに結合した．成果発表会に使用したスライドは，Google スライドで作成した．また，プロジェクトの概要や活動内容をまとめたポスターをプロジェクト全体で 2 枚作成した．使

用したポスターは Adobe　Illustrator で作成した．ポスターは 2 つのグループの内容が含まれているためここではなく付録 F で述べる．

（※文責: 八木田光）

2.5.2　発表内容

SWG の発表内容としては主に，目的・用いた器具の説明・データ取得・リアルタイムの可視化のデモンストレーションである．目的は 2.1 に記述されている通りである．用いた器具については，まずスマートウォッチ・活動量計が我々の身体に装着して身体データを取得することが可能であるデバイスであることを実際用いた器具を用意して説明した．データ取得に関しては，2.3.3 で記述されている内容に加え実際に ANT+ のドングルを用いて説明を行った．デモンストレーションでは，前期に SWG が 2.3 で述べられた内容で行ったようなリアルタイムでの心拍数の変動を確認した．また，評価フォームを設け発表後に聴講者に記入をお願いした．評価フォームは公立はこだて未来大学未来大学が用意したものであり，形式は Google form である．フォームの QR コードをスライド内に埋め込むことで聴講者に読み取ってもらった．

（※文責: 八木田光）

2.5.3　発表評価

中間発表会後評価フォームに寄せられたフィードバックを踏まえて反省を行った．フィードバックの内容に関しては SWG だけでなく ARG に該当するものも多く含まれているため，ここではなく 3.7.4 でまとめて述べる．

（※文責: 八木田光）

2.6　心拍数リアルタイム変動確認アプリケーション仕様書

ここでは，心拍数リアルタイム変動確認アプリケーションの仕様について述べる．

2.6.1　プログラム名

心拍数リアルタイム変動確認プログラム

（※文責: 八木田光）

2.6.2　開発環境

Windows 10 Home
Visual Studio Code (Java Script)
メモ帳 (html/css Windows 付属のテキストエディタ)

（※文責: 八木田光）

2.6.3 動作環境

Windows 64 bit

（※文責: 八木田光）

2.6.4 ディレクトリ構成

本アプリケーションにおけるフォルダのディレクトリ構造は以下のとおりである．モジュールファイルなどは記載を省略している．

- ディレクトリ構成

```
HeartBeats-master
  ├── packagelock.json
  ├── HeartBeats-master
  │  │  ├──.gitattributes
  │  │  ├──.gitignore
  │  │  ├── LICENSE
  │  │  ├── log.txt
  │  │  ├── main.js
  │  │  ├── package-lock.json
  │  │  ├── package.json
  │  │  ├── README.md
  │  │  ├── StartHeatBeats.bat
  │  │  ├── node_modules
  │  │  │  └──...
  │  │  ├── html
  │  │  │  ├── index.html
  │  │  │  ├── assets
  │  │  │  │  └──...
  │  │  │  ├── css
  │  │  │  │  ├── reset.css
  │  │  │  │  ├── style.css
  │  │  │  ├── js
  │  │  │  │  ├── jquery-3.3.1.min.js
  │  │  │  │  ├── main.js
  └──  ──  ──  ├── socket.io.js
```

これらのシステムのビルド済みフォルダは，本プロジェクトの Google ドライブ共有フォルダ（以下共有フォルダ，または project2022 と記載）内の以下のフォルダにある．

project2022/前期スマートウォッチグループ/HeartBeats-master

（※文責: 八木田光）

2.6.5 HeartBeats-master について

HeartBeats-master の主な機能は以下の通りであり，その概要を示す.

表 2.1 各スクリプトの主な機能

スクリプト名	概要
main.js	スマートウォッチ・活動量計との接続の確認および csv 出力
index.html	可視化を行うローカル html 内の配置・装飾
style.css	index.html のデザイン

各スクリプトの詳細を以下に記す.

main.js

このスクリプトは，ミドルウェア関数である app.get()，http.listen() と複数の on() メソッドによって構成されている.

app.get() では，__dirname で、現在実行中のソースコードが格納されているディレクトリパスを取得し，index.html を読み込む.

http.listen() では，設定されたポートでサーバを listen している.

io.on では，クライアント側で表示させるウェブページを作成するために，connection(webSocket 確立時) イベント時にクライアントが接続するたびにイベントハンドラ・id を登録

senor.on では，デバイス名が null または data.BeatCount と異なる場合にデバイス名を data.BeatCount にする.

io.emit でデバイス id・心拍数をサーバに送信．また，output.csv としてローカルファイルに出力.

stick.on では，スマートウォッチ・活動量計とドングルの接続の確認によって，コンソールの出力内容を定めている.

sensor.on では，センサーの付属の有無を確認している.

また，setInterval 以降ではコンソールでのデバイス id・心拍数・再起動までの秒数に表示形式を定めている.

以下のスクリプトについては文字や画像の配置についての内容となるため簡略して述べる.

index.html

タイトルは HeartBeats-viewer である．文字コードは UTF-8 となっている．ページの説明文は特に設けていない．また，外部リソースとして style.css を用いている．div は style.css の内容に基づいている．前述の script src は main.js や socket.io・jquery といったモジュールを指す.

style.css

#textArea では，心拍数の数値の配置に関するレイアウトを指定している.

#text では，心拍数の数値のサイズ・幅・色・フォント・不透明度を指定している.

#heartImage では，ハートの画像のサイズ・右側の余白・不透明度を指定している.

#chart では，グラフの位置・幅・周りの余白を指定している.

変数名	型	スコープ	説明
express	const	Global	express モジュールの読み込み
Ant	const	Global	ant-plus モジュールの読み込み
app	const	Global	読み込んだ express モジュールをインスタンスにする
http	const	Global	http サーバを立てる
io	const	Global	socket.io を用いてサーバに送信
PORT	const	Global	ポートの設定
timeout	const	Global	デフォルトの再起動までの時間
deviceId	let	Global	デフォルトのデバイス ID
deviceCount	let	Global	デバイスの名称
deviceBeat	let	Global	デバイスから取得した心拍数
timeoutCount	let	Global	ソフト再起動までの時間
stick	let	Global	Ant+ スティックのドライバ
sensor	let	Global	スマートウォッチ・活動量計における心拍計のセンサー
fs	var	Global	fs モジュールの読み込み
did	let	Local	id という文字列とデバイスの id を合わせたもの
timeoutCheckLoop	let	Local	未使用

（※文責: 八木田光）

2.6.6 動作の流れ

本アプリケーションの動作の流れを以下に示す.

スマートウォッチ・活動量計との接続

1. StartHeatBeats.bat を起動する.（これにより main.js が起動）
2. 設定されたポートでサーバに listen. その後ドングル起動を確認しコンソールに表示.
3. スマートウォッチ・活動量計のデバイス id を deviceCounts・心拍数を deviceBeat として取得し表示. また，取得したデータは fs.appendFileSync によって”output.csv”として StartHeatBeats.bat と同一のディレクトリに出力される. また，接続が確認されると同時に timeoutCount=10 として毎秒 timeCount が 1 減少. 0 になるとタイムアウトし StartHeatBeats.bat が再起動.

html での表示

1. StartHeatBeats.bat が起動された段階で　～ユーザーの任意のパス
   HeartBeats-master
   HeartBeats-master
   html にある index.html を手動で起動
2. src で指定した main.js や socket.io・jquery といったモジュールにより，リアルタイムの心

拍数がグラフとして変動．数値の文字の大きさ・グラフの色が css 内の指定によって表示.

（※文責: 八木田光）

第 3 章　AR アプリケーションの制作

この章では，前期に ARG が作成した成果物について述べる．

（※文責: 川上龍仁）

3.1　目的・目標

本グループは，当初計画していた「X-Reality とスマートウォッチを用いて身の回りの現象に接続し，自分の気持ちと世界をつなぐための情報リンクを作成する」というプロジェクト全体の最終目標達成のために，X-Reality を利用できるよう技術を習得するためのグループである．X-Reality とは，「Extended Reality」もしくは「Cross Reality」の略称で，VR（Virtual Reality），AR（Augmented Reality），MR（Mixed Reality），SR（Substitutional Reality）といった先端技術の総称である [6]．それぞれの技術の特徴は以下の通りである．

Virtual Reality（仮想現実）

VR は，仮想世界を現実のように体験できる技術である．CG や 360 度カメラによって作成された全方位の映像を専用のヘッドマウントディスプレイを装着して体験するため，どの方向に目を向けても，360 度の仮想空間を楽しめるのが特徴である．VR を体験する方法としては，「MetaQuest2（旧：OculusQuest2）」や HTC VIVE といったヘッドマウントディスプレイを利用するほか，スマホをセットするだけで利用可能な VR ゴーグルを利用する方法も存在する．また，手足にセンサーを装着し，体を動かすことでアバターを操作する技術も登場している．完全仮想世界ならではのアクション，視点，没入感がある一方で，通常動画の 2 倍データ容量が必要な点や，ヘッドマウントディスプレイ使用時に周囲の状況が分からなくなる点，視覚だけで情報を取得するので VR 酔いになりやすい点など様々なデメリットも存在している．

Augument Reality（拡張現実）

AR は，目の前に見えるリアルな現実の風景に様々な情報を付け加える技術で，大きく分けるとロケーションベース AR，マーカー型ビジョンベース AR，マーカーレス型ビジョンベース AR，SLAM(Simultaneous Localization and Mapping) の四つに分類される．スマホやヘッドマウントディスプレイを介して現実世界を見たときに，仮想の存在であるデータや画像を表示することで，現実世界を拡張することができる．AR と VR の相違点として，現実世界の映像の有無がある．VR では，仮想世界の映像のみが利用される．一方，AR では現実世界の映像の上に仮想世界の情報を重ねている．スマートフォンのようなモバイルデバイス上でも扱える手軽さと，その手軽さからコンテンツがシェアされやすいというメリットがある一方で，背景が明るすぎると視認性が落ちるデメリットもある．

Mixed Reality（複合現実）

MR は，現実世界の映像と仮想世界を融合する技術である．ヘッドマウントディスプレイ

のような MR 専用デバイスを装着することで，ユーザーの位置や動きに合わせたデジタル情報の表示・操作や，複数人での同時体験が可能である．AR との違いとして，AR は現実への情報の追加なのに対し，MR は仮想世界も現実世界同様に表現できる，「そこにいるかのような」をより精密に表現することを主目的としていることが挙げられる．立体映像での認識の容易さが特徴であるが，MR に対応しているスマートグラスを使用しなくてはいけないという扱いにくさもある．

Substitutional Reality（代替現実）

SR は，仮想世界を現実の世界に置き換え，あたかも今起きているかのように認識させる技術である．VR や AR では，体験者が目の前の仮想情報に対し「これは仮想情報だ」と気づくことができる前提があるが，SR の場合は体験者自身が SR を体験中である認識がありながらも，目の前に投影される仮想情報をほとんど現実の情報と同じ感覚で捉えてしまう．SR は VR,AR,MR より高性能である 360 度パノラマカメラ付きのヘッドマウントディスプレイと現実世界であらかじめ記録した映像や音声といった過去の情報が必要で，体験者は 360 度パノラマカメラ付きのヘッドマウントディスプレイを装着して現実世界をパノラマカメラ越しに現実世界をみる．2012 年に理化学研究所が開発した SR システムでは，過去の風景，人々といった映像を現在に置き換え，今まさに目の前で起きているかのような体験を提供することに成功した [7]．SR は未だ実験段階であり、実用化のアイデアも発展途上の段階で，さらに一般的な知名度も低い．そのため，具体的な事例はほとんどなく，現在では研究そのものがあまり行われていない．VR，AR，MR 以上に仮想の世界を自然に作り出すことが期待されている．

本グループはその中でも特に AR に注目し，四つの先端技術の中から AR を選択することに決定した．理由としては，デバイスを縛られることがないという手軽さと，現実世界と仮想世界の同時利用が「自分の気持ちと世界をつなぐ」に合致していると考えたためである．しかし，現在の本グループの状態では AR に関する技術や知識が不足しており，中間発表会までに最終成果物のプロトタイプを開発することは難しいと考えた．そのため，最終目標に AR を用いることができるように，前期活動では AR を用いたアプリケーション開発技術の習得を目的とした．具体的には，以下の二点を重視して学習を行った．

- Unity を用いたアプリケーション開発方法
- AR についての開発技術（AR の基礎技術であるタグ検知，平面検知など）

上記の目的を達成するため，中間発表会までに，習得した技術を成果物として表出することとした．具体的には，AR 開発用フレームワーク「AR Foundation」を用いたスマートフォンアプリケーションの制作を目標とした．これは，学習した技術をアプリケーションとして実装することができることによって，技術の習得を表すことができると考えたためである．本グループが前期活動において制作したアプリケーションは以下の 3 つである．

1. 平面検知アプリケーション
2. タグ検知アプリケーション
3. AR ダーツゲーム

平面検知アプリケーションでは，AR Foundation の一機能である平面検知と Prefab によるオブジェクトの生成を要素として導入した．タグ検知アプリケーションでは，こちらも AR Foundation の一機能であるタグ検知と平面検知同様 Prefab によるオブジェクトの生成を要素として導入した．AR ダーツゲームでは正確性の高かったタグ検知を利用したゲームを作成した．要素はタグ検知，Prefab によるオブジェクトの生成，オブジェクト衝突の当たり判定を導入した．　それぞれのアプリケーションの詳しい解説は，次のセクションより行われる．

（※文責: 川上龍仁）

3.2　平面検知アプリケーション

AR を用いたアプリケーションを制作するにあたり，「床や壁，机などの現実世界に存在する物体をカメラで認識し，それらを活用しアプリケーション側で対応させる」という仕様が何を制作するにしても必ず必要になると考えた．そのため，まずはじめに制作したのは，平面検知という技術（「3.6.1 平面検知」で詳しく後述）を利用したアプリケーションである．このアプリケーションは，床や壁などの平面を検知し，視認された平面をタップするとオブジェクトを表示するものである．アプリケーションの具体的な動作は以下の通りである．

1. スマートフォン内蔵のカメラで床や壁などの平面を検知する
2. 検知した平面を図のように黄色で可視化する
3. 可視化した平面がタップされると，その位置にオブジェクトを表示する

図 3.1　平面検知

図 3.2　可視化

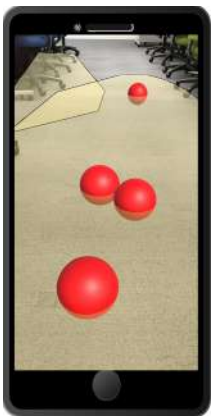

図 3.3　オブジェクト表示

完成した平面検知を利用したアプリケーションを使用した感想として，平面検知機能に限界があると感じた．というのも，今回利用した平面検知機能は “AR Foundation” という AR 開発用フレームワークの一機能なのだが，うまく検知できない状況が発生することがわかった．明るい部屋の一般的な床等なら問題なく検知することができ可視化も正確に行うことができていたのが，明暗差のある部屋であったり，光が差し込んでいる場所，複数の色がタイル状に並んでいる模様の床などは平面を正確に認識することができな場合があった．このことから，AR Foundation の平面検知機能を実際に使用していくことは難しいと判断した．しかし，アプリケーション内のタップ処理やカメラの使用法など学べる内容も多かったので，総じて有用なアプリケーション制作であった．

（※文責: 川上龍仁）

実装手順

1.「GameObject」→「Cube」を選択する．
2.「Cube」を選択し，「Scale」の X,Y,Z を 0.5 に設定する．
3.「Cube」をプロジェクトウィンドウの「Assets」の中にドラッグアンドドロップし，Prefab 化する．
4. Hierarchy から「Cube」を削除する．
5.「AR Session Origin」を選択し，「Add Component」をクリックする．
6.「AR」で検索し，「AR Plane Manager」「AR Raycast Manager」を追加する．
7.「GameObject」→「AR Default Plane」を選択し，手順 2,3 と同様に Prefab 化する．
8.「AR Plane Manager」の「Plane Prefab」に手順 6 で作成した「AR Default Plane」をドラッグアンドドロップする．
9.「Assets」→「Create」→「C# Scripts」を選択する．
10. スクリプト名を「SpawnCube」に変更し，スクリプト（付録 B.1.1 に記載）の内容を反映させる．
11.「AR Session Origin」を選択し，「SpawnCube」を Inspector にドラッグアンドドロップする．
12. スクリプトの「Cube」欄に「Cube」の Prefab をドラッグアンドドロップする．

（※文責: 川上龍仁）

3.3　タグ検知アプリケーション

上記の問題をもとに，本グループは平面検知機能の改善をする，もしくは別の方法で床や壁など現実世界の物体をカメラで認識できないかを検討した．そこで，検知精度を上げるための方法として AR タグというものを活用したタグ検知（「3.6.2 タグ検知」で詳しく後述）という方法を試してみることにした．このタグ検知も平面検知同様 AR Foundation 内包の一機能だが，内容が異なるため挑戦する価値はあると考えた．このアプリケーションは，事前に登録した画像（AR タグ）を検知すると，そのタグに対応したオブジェクトを表示するものである．具体的な動作は以下の通りである．

1. スマートフォンのカメラで AR タグを検知する
2. 検知したタグの位置にオブジェクトを表示する

図 3.4　タグ検知

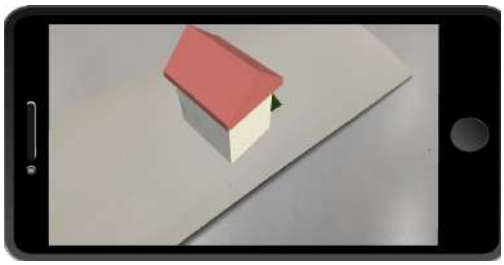

図 3.5　オブジェクト表示

完成したタグ検知を利用したアプリケーションを使用してみるととても使い勝手が良かった．も

ともと平面検知の代用案として制作していたのだが，想定以上に正確で優秀だったため，今後の方針として AR タグを利用したタグ検知を活用していく方向で進めることになった．ただしまだ完全には習得できていなかったので，複数の AR タグの同時利用や異なるオブジェクトの出現など本番で使える技術を覚える必要はあると感じた．これらのことから，現実世界の認識は平面検知よりもタグ検知の方が正確で利用しやすいためタグ検知を使用することと，それをうまく活用できる成果物のアイデアを考えることへ方針を定めることができた．

（※文責: 川上龍仁）

実装手順

1.「GameObject」→「Cube」を選択する．
2.「Cube」を選択し，「Scale」の X,Y,Z を 0.5 に設定する．
3.「Cube」をプロジェクトウィンドウの「Assets」の中にドラッグアンドドロップし，Prefab 化する．
4. Hierarchy から「Cube」を削除する．
5. タグとして登録する画像を用意し，プロジェクトウィンドウの「Assets」の中にドラッグアンドドロップする．
6. プロジェクトウィンドウの「Assets」の中で右クリックし，「Create」→「XR」→「Reference Image Library」を選択する．
7.「Reference Image Library」を選択し，Inspectot の「Add Image」をクリックする．
8. タグとなる画像を正方形の枠内にドラッグアンドドロップする．
9.「Specify Size」にチェックを入れ，画像の実物サイズ（メートル単位）を入力する．
10.「AR Session Origin」を選択し，Inspector の「Add Component」をクリックする．
11.「AR」で検索し，「AR Tracked Image Manager」を追加する．
12.「Serialized Library」の欄に，「Reference Image Library」をドラッグアンドドロップする．
13.「Max Number of Moving Images」に 1 を入力する．
14.「Tracked Image Prefab」の欄に「Cube」をドラッグアンドドロップする．

（※文責: 川上龍仁）

3.4　AR ダーツゲーム

これまで制作してきたアプリケーションや会得した技術の総まとめとして，AR ダーツアプリケーションを制作することとした．具体的には，AR タグを用いたタグ検知，オブジェクト生成，当たり判定といった機能をすべて導入することを条件としたうえでゲームとして成立させるアプリケーションの制作となる．これまでの二つのアプリケーションと比べて総量が多いため難易度は上がるが，グループでの分担などを心がけ協力して取り掛かることとした．このアプリケーションは，AR 空間上の的に向かって矢を投射し，的に当たった場合に矢をその場で停止させるものである．的の表示には，タグ検知を用いている．具体的な動作は以下の通りである．

1. スマートフォンのカメラを用いて AR タグを認識する

タグ検知機能を用い，事前に登録した任意の画像を AR タグとして認識することができ

る．発表の際にはフリーアイコンを採用したが，いつでも特定の画像に変更することができる．

2. 検知した AR タグの位置に，ダーツの的となるオブジェクトを表示する

表示するダーツの的は 3DCG 制作ソフト「Blender」で制作したものを使用した．

3. 画面がタップされると，画面中央から矢のオブジェクトを投射する

矢の投射は本数指定が無く，タップする度に生成されるため何本でも挑戦することができる．

4. 投射した矢が的に命中すると矢が静止する

当たり判定機能を用い，矢が的に当たった際に当たったことが分かるよう矢を静止させた．

図 3.6　AR ダーツゲーム

前期の成果物として発表・提出したこの AR ダーツアプリケーションは，まだ改善の余地はあるものの，前期目標の達成としては十分な完成度を持っていたと言える．アプリケーション制作に用いた技術はもちろん，グループ開発の流れや，Unity や Git といった開発プラットフォームの基礎的使い方，スケジュールの組み方など様々なことを知り能力を培うことができた．これらも技術習得の一部として広く捉え，多くのものを習得することができた前期であったと考える．

（※文責: 川上龍仁）

実装手順

1. ダーツの矢と的のモデルを用意し，Prefab 化する（本プロジェクトでの実装の際は，Blender を用いて的と矢を作成した）．
2. ダーツの矢の Prefab 名を「Allow_c」，的の Prefab 名を「Plate」とする．
3.「Allow_c」を選択し，Inspectot の「Add Component」をクリックする．

4.「Rigidbody」で検索し，「Rigidbody」を追加する．
5. 3.3 の実装手順でタグ検知アプリケーションを作成する．
6.「AR Tracked Image Manager」の「Tracked Image Prefab」の欄に，「Plate」Prefab をドラッグアンドドロップする．
7.「Assets」→「Create」→「C# Scripts」を選択し，スクリプトを 2 つ作成する．
8. スクリプト名をそれぞれ「Dart」「CapsuleScript02」に変更し，スクリプト（付録 B.2.1 に記載）の内容を反映させる．
9.「Allow_c」Prefab を選択し，「Dart」スクリプトを Inspectot にドラッグアンドドロップする．
10.「AR Session Origin」の内部にある「AR Camera」を選択し，「CapsuleScript02」スクリプトを Inspectot にドラッグアンドドロップする．

（※文責: 小田涼平）

3.5 開発環境

AR アプリケーション制作時の開発環境を以下に示す（本環境の構築手順は「3.9 環境構築」に記述）．

表 3.1 開発環境一覧

開発プラットフォーム	Unity 2021.3.3f1
AR 開発フレームワーク	AR Foundation 4.2.3
ビルド対象プラットフォーム	Android 11
バージョン管理ツール	GitHub

Unity

Unity とは，ユニティ・テクノロジーズ社が提供する，ゲーム開発プラットフォームである．今回は，AR アプリケーションの開発と，開発したアプリケーションのビルドを行う目的で使用した．AR アプリケーション開発には，Unity が提供する AR 開発用フレームワーク「AR Foundation」を使用した．本グループでは，AR Foundation で提供されている機能のうち，平面検知機能 [8] とタグ検知機能 [9] についての学習を行った．開発したアプリケーションのビルドには，Android スマートフォン “AQUOS sence 4” をテスト端末として使用した．なお，Android アプリケーションのビルドには，Android Keystore Manager から Keystore を作成する必要があった．

（※文責: 小田涼平）

GitHub

GitHub は，CUI ツール（コマンドラインツール）の一種であり，ソースコードのバージョンをいつだれがどこを編集したのか，最新のバージョンはどれになるのかなどを管理するツールである [13]．本プロジェクトでは，開発する様々な Unity プロジェクトをメンバー間で共有し，バージョンを管理する目的のために使用した．Unity プロジェクトを GitHub で管理する際の注意点

として，Unity プロジェクトはファイル容量が大きすぎる点がある．もし Unity プロジェクトをそのまま GitHub に保存してしまうと，個人で使える Git 容量が一瞬にして最大になってしまう．そのため，対策として「.gitignore ファイル」を使用し，GitHub で管理するファイルを制限した．.gitignore とは，追跡する必要のないファイルをあらかじめ設定しておくことで Git が追跡しなくなるという機能である．この.gitignore を使用する際の注意として，.gitignore ファイルは Unity プロジェクトフォルダの直下に配置する必要があった．それ以外の配置をして．gitignore が上手く作動しない場合が確認された．

（※文責: 小田涼平）

3.6 使用した技術

この節では，上記アプリケーションを制作する際に使用した技術の概要と，大まかな実装方法について述べる．

（※文責: 小田涼平）

3.6.1 平面検知

概要

平面検知とは，カメラで取得した画像内から，床や壁などの平面を検知することができる技術である．AR Foundation では，AR Plane Manager というコンポーネントにより平面検知機能が提供されている．

（※文責: 小田涼平）

アプリケーションでの実装方法

平面検知機能は，「3.2 平面検知アプリケーション」において使用された．本機能の大まかな実装手順を以下に示す．

1. AR Foundation パッケージをインストールする．
2. ヒエラルキー上に AR Session と AR Session Origin を追加する．
3. AR Session Origin に AR Plane Manager コンポーネントを追加する．

（※文責: 小田涼平）

3.6.2 タグ検知

概要

タグ検知とは，カメラで取得した画像内から，認識対象となる画像を検知することができる技術である．この認識対象となる画像は，「AR タグ」や「AR マーカー」などと呼ばれる（以下，AR タグと呼ぶ）．AR Foundation では，AR Tracked Image Manager というコンポーネントによりタグ検知機能が提供されている．また，AR タグは Reference Image Library というライブラリに登録する．

（※文責: 小田涼平）

アプリケーションでの実装方法

タグ検知機能は，「3.3 タグ検知アプリケーション」と「3.4 AR ダーツゲーム」において使用された．本機能の大まかな実装手順を以下に示す．

1. AR Foundation パッケージをインストールする．
2. ヒエラルキー上に AR Session と AR Session Origin を追加する．
3. プロジェクトウィンドウ上に Reference Image Library を追加する．
4. Reference Image Library に AR タグを登録する．
5. AR Session Origin に AR Tracked Image Manager コンポーネントを追加する．
6. AR Tracked Image Manager の Serialized Library に，Reference Image Library を追加する．

（※文責: 小田涼平）

3.6.3 Prefab によるオブジェクト生成

概要

オブジェクトの生成には，Unity の機能である Prefab[10] を使用した．Prefab 化したオブジェクトは，再利用可能なアセットとして保存される．オブジェクトを Prefab 化することにより，同一のオブジェクトを複数個生成することが容易になる．

（※文責: 小田涼平）

アプリケーションでの実装方法

Prefab によるオブジェクト生成は，「3.2 平面検知アプリケーション」，「3.3 タグ検知アプリケーション」と「3.4 AR ダーツゲーム」において使用された．本機能の大まかな実装手順を以下に示す．

1. ヒエラルキー上に Prefab 化したいオブジェクトを生成する．
2. 生成したオブジェクトをプロジェクトウィンドウにドラッグアンドドロップする．
3. ヒエラルキーから生成したオブジェクトを削除する．

（※文責: 小田涼平）

3.6.4 当たり判定

概要

オブジェクトどうしが衝突したかどうかの判定を行う際には，OnCollisionEnter 関数 [11] を使用した．OnCollisionEnter は，オブジェクトどうしの衝突を検知すると呼び出される関数である．そのため，衝突を検知した際の処理を OnCollisionEnter 関数の中に記述することで，当たり判定を実装できる．

（※文責: 小田涼平）

アプリケーションでの実装方法

当たり判定の機能は，「3.4 AR ダーツゲーム」において矢が的に刺さる挙動を実装するために使用された．矢のオブジェクトに Rigidbody という物理演算を扱うコンポーネントを追加している．そして，矢が的に当たると Rigidbody のパラメータである isKinematic[12] を有効にして，物理演算の影響を受けないようにしている．これにより矢が的に当たると，矢が的の上で静止しているようにみえる．本機能の大まかな実装手順を以下に示す．

1. 矢のオブジェクトに Rigidbody コンポーネントを追加する．
2. スクリプトの OnCollisionEnter 関数内で，矢の isKinematic を true にする．

（※文責: 小田涼平）

3.7　中間発表会

ここでは，前期に行った中間発表会（2022,7,8）の内容のうち前期 ARG に関する部分を記述する．

（※文責: 川上龍仁）

3.7.1　発表準備

中間発表に向けた準備として，成果物以外の発表に必要なスライドとポスター，原稿，評価フォームの制作を行った．

発表スライド

発表スライドは，SWG と ARG で成果物が異なるためそれぞれで成果物紹介や導入といった内容のスライドを作成し，完成したのちに統合させた．スライドの制作は，「Google スライド」を用いた．Google スライドとは，Google 社が提供するウェブ上でプレゼンテーション資料を作成することができるツールである．統合は，前半を SWG，後半を ARG として，発表の順番も固定できるよう行った．

発表ポスター

発表ポスターは，合計で二枚制作した．一枚目には本プロジェクトの概要，SWG と ARG の各目標と成果物の端的な紹介，後期に向けた今後の展望を記した．二枚目にはそれぞれのグループの少し詳細な成果物説明と，使用した技術の一例などを記した．発表ポスター制作の意図としては，発表開始を待つ間の軽い予習と発表を聞いた後に内容を簡単に振り返ることができるからである．これによって，発表内容を理解しやすくなる，発表後の質問が容易になるなどの利点がある．

発表原稿

発表原稿は，SWG と ARG それぞれのグループで制作した．発表原稿は，発表スライドを制作した Google スライド内のスピーカーノートに記述し活用した．基本的に発表原稿は各自暗記し，スピーカーノートを確認しなくても説明が出来る状態になるようにしている

が，本番での緊張や焦りによって内容を忘れてしまった際の措置として配置した．

評価フォーム

評価フォームは中間発表に参加し本プロジェクトの発表を聞いていただいた方達を対象にした，発表内容や態度を段階評価できるサイトの事である．評価フォームは Google 社の Google フォームを活用し，QR コード読み取りでのアクセスを可能とした．なおこの評価フォームに用意された各評価項目は，すべて未来大 WG が用意しているものであり，本プロジェクトはその形式を借りて QR コードを作成し使用した．

（※文責: 川上龍仁）

3.7.2 発表内容

ARG の発表内容として，集大成として最後に制作した AR ダーツゲームの紹介を中心とした学習成果の発表と後期に向けた展望を述べた．技術習得という目的のための成果物制作であること，学んだ技術を活用するのに AR ダーツゲームが適任であったこと，そのほか学んだ技術を端的に分かりやすく画像を交えながら説明をし，最後に AR ダーツゲームの実演・体験をその場で行った．AR ダーツゲームの実演では，実際にゲームを取り込んだスマートフォンを用意し，画面を見せながら遊んで見せた．その後，聴衆から一・二名ほど抜擢し，スマートフォンを渡し操作させた．これによって聴衆は，説明だけでは分かりにくい部分などを経験によって感じられたと考えられる．

（※文責: 川上龍仁）

3.7.3 評価シートについて

評価フォームには，評価者の種別や，学籍番号または所属，評価者指名，発表技術の 10 段階評価，発表技術についてのコメント，発表内容の 10 段階評価，発表内容についてのコメントの 7 項目があった．

中間発表会後に集計した評価シートの回答総数は 43 件，うち 3 件が教員からの回答であった．

（※文責: 八木田光）

発表技術についてのコメント

平均点は 6.9 点，中央値は 7 点であった．また，コメントのうち高評価であったもの（8 点以上）を一部抜粋したものを，以下にまとめる．

- とてもスムーズに発表しており良かったと思う
- 声が大きく聞きやすかった．スライドの内容が分かりやすくまとめられていた．
- ほぼ全ての実験内容で得ることのできた結果を動画にしてくれていたのでわかりやすく発表者の話に入り込みやすかったです．
- 実演などを交えて興味をひく発表は良かったと思います．
- 実演や動画などの工夫があり，飽きさせない工夫があった．

次に，コメントのうち低評価であったもの（6 点以下）を一部抜粋したものを，以下にまとめる．

- 周りのプロジェクトが大きかったので，少し声が聞き取りにくかった気がする．実物を見せたりするのが良かった．
- 体育館は広いので，声があまり響いていなかった．
- 具体例を述べて分かりやすく説明している声が少し小さかった
- 初めのスライドの時声の大きさがもう少しあると良いと思った．マイクとか使っても良いと思う．
- 発表はもっと前を見て話せると聞き取りやすいのかなと思った．前半は問いかけを入れるなど聴衆の意識を向けさせる工夫があるのがいいと思った．
- 声が聞きづらい．資料を見ていないのはいい．スライドの字数が少なくて理解しづらいように感じた．映像も使われていてわかりやすい．
- 一人目の人は良かったけど，二人目の人が，スマホの台本を全集中で読み上げていて残念だった．スクリーンから 5m 離れて指差ししていた．
- グループ間でスライドの雰囲気が統一されていて良かった．声の大きさにばらつきがあるので，改善できそうです．

（※文責: 八木田光）

発表内容についてのコメント

平均点は 7.5 点，中央値は 8 点であった．また，コメントのうち高評価であったもの（8 点以上）を一部抜粋したものを，以下にまとめる．

- マーカーを使用する AR ではなくマーカーレス AR を用いることによってユーザの利便性を向上させることができると思う．心拍を取得できており，AR ダーツゲームも作成できていることから進捗及び目標設定は適切であると思われる．
- 画像，実演などを使って分かりやすい説明だった．
- 心拍数を実際に用いてくれたのでとてもわかりやすかったです．
- Unity 側の自由度でたぶん何でもできるので頑張って下さい．応援してます．
- 説明は簡潔で分かりやすかった．質疑も的確に答えてくれた．
- 心拍数で気持ちを表現するというのは，リラックス状態や怒っているのは心拍数の上がりと下がりで判断出来ると直感的にわかるが，嬉しいなどの他の気持ちをどのように表現するのでしょうか？
- これから作るゲームについて，まだ具体的な案がハッキリ出てなさそうに感じました．扱っているコンテンツは面白いと思ったので期待しています．

（※文責: 八木田光）

3.7.4 発表評価と反省

中間発表後は，募っていた評価フォームに寄せられた段階評価，各コメントなどのフィードバックを基に，プロジェクト全体で反省を行い，後期に向けた改善策などを考えた．発表技術については，声が聞きやすいというコメントと聞きづらいというコメントの両方が見られた．このことから，発表者によって声の大きさにばらつきがあるという反省をし，最終発表では発表の声量を最も意識するように気をつけた．また，発表スライドだけでなく，動画を使用したり実演を交えた点は

高評価であった．後期も我々の取り組みを伝えやすくし，興味を持ってもらうために引き続き様々な形式を取り入れていくことにした．

発表内容については，心拍数をリアルタイムで扱うことや，Unity を用いたゲーム作成に関しては興味・関心を持ってもらうことができた．それと同時に，心拍数と気持ちの関連性や今後の展望の不透明さに関する指摘が多くみられた．これらを含めて，プロジェクトとしてリアルタイムで取得した心拍数をどう活用していくか，2 つのグループの前期の活動を含めて後期どのような課題設定を行っていくかという議論の必要性を確認した．

（※文責: 八木田光）

3.8 AR ダーツゲーム仕様書

3.8.1 概要

本プロジェクトにおける AR ダーツゲームは，AR グループの開発技術向上を目的として制作された Android アプリケーションである．AR ダーツゲームの大まかな動作手順を以下に示す．

1. スマートフォンのアウトカメラで AR タグを認識する
2. AR タグの位置に，ダーツの的となるオブジェクトが生成される
3. 画面をタップすると，画面中央から矢のオブジェクトが投射される
4. 投射した矢が的に命中すると矢が静止する

（※文責: 森皐太）

3.8.2 開発環境

本アプリケーションの開発環境を以下に示す（開発環境の構築手順は 3.9 を参照）．

表 3.2 開発環境一覧

開発プラットフォーム	Unity 2021.3.3f1
AR 開発フレームワーク	AR Foundation 4.2.3
ビルド対象プラットフォーム	Android 11
ビルド用端末	AQUOS sense 4
バージョン管理ツール	GitHub

（※文責: 森皐太）

3.8.3 ディレクトリ構成

本アプリケーションにおける Unity プロジェクトフォルダのディレクトリ構造は以下のとおりである．システムファイルなどは記載を省略している．

- ディレクトリ構成

```
ar-MarkerAndShoot
├── Assets
│   ├── Marker
│   │   ├── kitune.jpg
│   │   ├── ReferenceImageLibrary.asset
│   │   └── usi.jpg
│   ├── Materials
│   ├── Prefab
│   │   ├── Arrow_c.prefab
```

```
|  |  └── Plate.prefab
|  ├── Scenes
|  ├── StreamingAssets
|  ├── CapsuleScript01.cs
|  ├── CapsuleScript02.cs
|  ├── Dart.cs
|  └── XR
|  └── Loaders
|  └── Settings
├── Packages
└── ProjectSettings
```

（※文責: 森皐太）

3.8.4　オブジェクト構成

ヒエラルキーの構造は以下のとおりである．

- ヒエラルキー構成

```
SampleScene
  ├── Directional Light
  ├── AR Session Origin
  |  └── AR Camera
  └── AR Session
```

Directional Light

光源の役割を持つライトオブジェクトである．

AR Session Origin

スマートフォンのカメラで検出した平面などの特徴を，Unity 空間の座標や向きなどに変換するオブジェクトである．このオブジェクトには，以下のコンポーネントを追加している．

- AR Tracked Image Manager（タグ検知）

AR Camera

AR 上でカメラの役割をするオブジェクトである．AR Sesion Origin の子オブジェクトとして配置される．このオブジェクトには，以下のコンポーネントを追加している．

- CapsuleScript02.cs（後述）

AR Session

AR 上での処理を制御するためのオブジェクトである．

（※文責: 森皐太）

3.8.5　使用プレハブ

本アプリケーションで使用したプレハブは以下のとおりである．

表 3.3　使用プレハブ一覧

プレハブ名	概要
Arrow_c.prefab	ダーツの矢オブジェクト
Plate.prefab	ダーツの的オブジェクト

使用したプレハブの詳細を以下に示す．

Allow_c.prefab

ダーツの矢を模したオブジェクトである．また，このプレハブには以下のコンポーネントを追加している．

- Rigidbody（物理演算）
- Dart.cs（後述）

Plate.prefab

ダーツの的を模したオブジェクトである．また，このプレハブには以下のコンポーネントを追加している．

- Rigidbody（物理演算）

（※文責: 森皐太）

3.8.6　AR タグの検知設定

AR タグの検知には，前述した AR Tracked Image Manager コンポーネントを使用した．そして AR タグを検知した場合，ダーツの的のプレハブ（Plate.prefab）を AR タグの位置に生成するように設定した．AR タグのライブラリには，Reference Image Library を使用した．また，AR タグとなる画像は，ウシのイラスト（usi.jpg）とキツネのイラスト（kitune.jpg）の 2 つを設定した．

図 3.7　使用したウシのイラスト

図 3.8　使用したキツネのイラスト

（※文責: 森皐太）

3.8.7　プログラム構成

本アプリケーションにおいては，以下のスクリプトを作成した．

表 3.4　スクリプト一覧

スクリプト名	概要
CapsuleScript01.cs	タップ動作による矢の投射（未使用）
CapsuleScript02.cs	タップ動作による矢の投射
Dart.cs	的に命中した際に矢を静止

各スクリプトの詳細を以下に示す．

CapsuleScript01.cs

このスクリプトでは，タップ動作を検知すると Allow_c プレハブから矢のオブジェクトを生成し，画面中央から矢を投射する挙動を実装している．

画面中央から矢を投射する方法として，Camera.main.ScreenToWorldPoint 関数を用いてスクリーンの中央の座標を求め，その中央座標から矢を投射する方式を用いている．なお，本スクリプトよりも次に記述する CapsuleScript02 の方が画面中央から投射する挙動を単純に実装できたため，本アプリケーションにおいて本スクリプトは使用しなかった．

CapsuleScript02.cs

このスクリプトでは，タップ動作を検知すると Allow_c プレハブから矢のオブジェクトを生成し，画面中央からその矢オブジェクトを投射する挙動を実装している．

タップ動作の検知には，Input.touchCount を用いている．Input.touchCount が 1 以上になった場合，矢を生成し投射する処理を行う．

画面中央から矢を投射する方法として，矢を AR カメラの子オブジェクトとして生成する方式を用いている．これにより，スマートフォンのカメラの向きを変えても，矢は画面中央（AR カメラの座標）から投射される．

矢のオブジェクト生成には，Instantiate 関数を用いている．生成対象のオブジェクトには，Allow_c.prefab を指定している．生成する位置と向きは，Allow_c.prefab の位置（transform.position）と向き（transform.rotation）を指定している．

矢を投射する挙動は，AddForce 関数を用いて前方に力を加えることにより実現している．なお，AddForce 関数の ForceMode は Impulse を指定している．

表 3.5　変数一覧とその説明

変数名	型	スコープ	説明
Allow_c	GameObject	Global	矢のプレハブを指定するオブジェクト
force	float	Global	矢に加える力の大きさを調整する変数
dart	GameObject	Local	タップの度に生成される矢のオブジェクト

Dart.cs

このスクリプトでは，矢が的に命中した際に矢を静止させる挙動を実装している．OnCollisionEnter 関数で衝突を検知した場合，衝突先のオブジェクトが的かどうか調べるためにオブジェクト名を参照する．衝突先のオブジェクト名が Plate（的のオブジェクト）

だった場合，矢の Rigidbody コンポーネント内の isKinematic を true に設定する．これにより，矢の衝突を検知した場合，CapsuleScript02 の AddForce 関数で加えた力を止め，矢の動きを静止させている．

（※文責: 森皐太）

3.8.8 動作の流れ

本アプリケーションの動作の流れを以下に示す．なお，ダーツの的の表示とダーツの矢の投射を行う順序は問わない．

ダーツの的の表示

1. AR Tracked Image Manager で AR タグ（Referenece Image Library で登録したウシまたはキツネのイラスト）を検知する．
2. AR タグの座標にダーツの的（Plate.prefab）を生成する．

ダーツの矢の投射

1. 画面がタップされると，Input.touchCount が 1 以上になる．
2. Instantiate 関数により，ダーツの矢（Arrow_c.prefab）を画面中央（タップされた時点での AR カメラの位置と向き）に生成する．
3. Addforce 関数により，ダーツの矢（Arrow_c.prefab）の前方の向きに Impulse モードで力を加え，ダーツの矢を投射する．

矢の命中・静止

1. ダーツの的を表示している状態で，的に向かってダーツの矢の投射が行われる．
2. OnCollisionEnter 関数により，矢の衝突を検知する．
3. 衝突先のオブジェクト名を参照する．
4. 参照したオブジェクト名が Plate だった場合，衝突した矢オブジェクトの Rigidbody コンポーネント内の isKinematic を true に設定し，矢オブジェクトを静止させる．

（※文責: 森皐太）

3.9 環境構築

Unity の環境構築手順

ここでは，Windows 版 Unity 2021.3.3f1 の環境構築手順を説明する．

（※文責: 森皐太）

3.9.1 Unity Hub のインストール

1. ダウンロードページ（`https://unity.com/ja/download`）にアクセスする．
2.「Windows 用ダウンロード」をクリックし，インストーラー（UnityHubSetup.exe）をダウンロードする．
3. ダウンロードが終了したら実行する．
4. インストーラーの手順に従い，インストールする．
5. Unity Hub を実行する（Unity エディターをインストールする画面が開かれた場合，「Skip installation」をクリックする）．

（※文責: 森皐太）

3.9.2 Unity 2021.3.3f1 のインストール

1. ダウンロードページ（`https://unity.com/releases/editor/archive`）にアクセスする．
2.「Unity 2021.3.3」の欄にある「Unity Hub」をクリックし，Unity Hub を開く．
3.「Install」をクリックする．

（※文責: 森皐太）

3.9.3 Android 開発環境の構築

AR アプリケーションの開発にあたり，以下の手順で Android 開発環境を構築した．

Android 用開発モジュールのインストール

1. Unity Hub を開き，「Installs」をクリックする．
2.「2021.3.3f1」の歯車をクリックし，「Add module」をクリックする．
3.「Android Build Support」を選択する（この際，「OpenJDK」，「Android SDK & NDK Tools」にもチェックが入っているか確認する）．
4.「Continue」をクリックし，モジュールをインストールする．

（※文責: 森皐太）

Keystore の作成

1. Unity のプロジェクトを開く（プロジェクトを作成していない場合は新規作成する）．
2.「File」→「Build Settings」を選択する．
3.「Android」を選択し，「Switch Platform」をクリックする．
4.「Edit」→「Project Settings」を選択する．
5.「Player」を選択し，「Publishing Settings」をクリックして展開する．
6.「Keystore Manager」をクリックする．
7. 左上のタブをクリックし，「Create New」→「Anywhere」を選択し，ファイルの保存場所とファイル名を指定する．

8. 以下の項目を入力する.
    - Password
    - Confirm Password（確認用のパスワード）
    - Alias
    - Password
    - Confirm Password（確認用のパスワード）
    - First and Last Name
9.「Add key」をクリックし,「Yes」をクリックする.

（※文責: 森皐太）

3.9.4 AR Foundation 開発環境の構築

AR アプリケーションの開発にあたり，以下の手順で AR Foundation の開発環境を構築した.

AR プラグインのインストール

1. 新規 3D プロジェクトを作成する.
2.「Window」→「Package Manager」を選択する.
3.「Features」欄の「AR」を選択し,「Install」をクリックする.

（※文責: 森皐太）

XR Plug-in Management の設定

1.「Edit」→「Project Settings」を選択する.
2.「XR Plug-in Management」を選択し，Android タブの「ARCore」にチェックを入れる.

（※文責: 森皐太）

Android ビルド設定

1.「File」→「Build Settings」→「Player Settings」を選択し,「Other Settings」を展開する.
2.「Auto Graphics API」のチェックを外す.
3.「Graphics API」の「Vulkan」を選択し，マイナスボタンを押す.
4.「Minimum API Level」を Android 7.0 にする.

（※文責: 森皐太）

AR Session と AR Session Origin の追加

1.「GameObject」→「XR」→「AR Session」を選択する.
2.「GameObject」→「XR」→「AR Session Origin」を選択する.
3. Hierarchy 上の Main Camera を右クリック→削除する.

（※文責: 森皐太）

GitHub の利用手順

GitHub と GitHub Desktop の環境構築手順，使い方，そして GitHub Desktop による Unity の管理方法について説明する．

（※文責: 宮崎隼）

3.9.5 GitHub を使うための準備

GitHub アカウントの作成

1. GitHub のページ（`https://github.com/`）を開く．
2. 登録用メールアドレスを入力し，「Sign up for GitHub」ボタンをクリックする．
3. メールアドレス，パスワード，ユーザー名を設定しすべてを終えたら「Create account」をクリックする．
4. 画面に「Welcome to GitHub」が表示されたら，下の方へスクロールし「Complete setup」ボタンをクリックする．
5. 画面に「Please verify your email address」が表示され，登録したメールアドレス宛に GitHub からメールが届くのを確認する．
6. メールの「Verify email address」ボタンをクリックする．
7. GitHub 画面上で「Your email was verified」メッセージが表示されていることを確認する．
8. 画面の「skip this for now」ボタンをクリックする．

（※文責: 宮崎隼）

プロジェクト全体の共有設定

プロジェクトの一人が行う．主に次のサイトを参考に行った [14].

1. GitHub ページの右上の [+] アイコンをクリックする．
2. 画面上の「New organization」をクリックする．
3. 料金プランの選択画面に遷移し，「Free」の「Join for free」をクリックする (プランは後でも変更可)．
4. Organization 名，代表者のメールアドレスを入力し，「My personal account」を選択して「Next」をクリックする．
5. 画面上に「Welcome to GitHub」のページが開き，アンケートを答え（アンケートは今後に影響なし)，「Submit」をクリックする．
6. 画面上に「Welcome to （Organization 名)」のページが開かれ，「Complete setup」をクリックする．
7. Organization が作成され，トップページが表示される．
8. 画面上の「People」タブをクリックして，「Invite member」をクリックして，プロジェクトメンバーの GitHub アカウント名を入力して「Invite」をクリックする．
9. 画面上の「Member」を選択して「Send invitation」をクリックする．

補足 1 もしプロジェクトの費用で有料プランにするのならば担当教諭に相談すること．

補足 2 プランの内役はこちらに->（https://github.co.jp/pricing.html）

（※文責: 小田涼平）

メンバーの登録

1. 招待されたメンバーはメール内の「Join @（Organization 名)」をクリックする.
2. GitHub のページが開かれ,「Join（Organization 名)」をクリックする.
3. GitHub ページの右上の [+] ボタンをクリックし,「Your organization」をクリックして, Organization 名をクリックして所属していることを確認する.

これにより個人アカウントに各ユーザーはサインインし，複数の個人アカウントが同じ Organization アカウントに参加すると，共有プロジェクトで共同作業を行うことができる.

（※文責: 小田涼平）

リポジトリの作成手順

1. GitHub ページの右上の「+」アイコンをクリックする.
2.「New repository」をクリックする.
3. 画面が「Create a new repository」に遷移され,「Owner」のプルダウンから作成した Organization 名を選択する.
4. リポジトリ名を「Repository name」に入力し,「Description」でリポジトリの説明を入力する.
5. 選択画面の「Private」にチェックする.
6. 画面下の「Initialize this repository with a README」にチェックする.
7. 画面下の「Add a license」欄は「MIT License」を選択する.
8. 画面下の「Add . gitignore」は Unity や Android studio を使うのならば C++ としておく.
9. 画面下の「Create repository」をクリックする.

この方法以外の「3.9.7 Unity プロジェクトを GitHub Desktop で管理する方法」のグループの一人が行う箇所でも説明している．そちらの方では GitHub Desktop からリポジトリの作成をし，なおかつ，Unity プロジェクトの管理も行えるものとなっている．Unity を扱う場合ならば GitHub Desktop の方が比較的容易に作ることができる.

（※文責: 小田涼平）

メンバーの登録

主に次のサイトを参考に行った [15].

1. 画面右の「Settings」をクリックする.
2. 画面左から「Collaborators」を選択する.
3. 画面に「Confirm password to continue」が表示され，GitHub アカウントのパスワードを入力して「Confirm password」をクリックする.
4.「Manage access」の中にある「Add people」をクリックする.
5.「search by username, full name or email address」欄にグループメンバーの GitHub アカウントかメールアドレスを入力し,「Add （ユーザー名） to this repository」をクリック

する.

補足 1 もしプロジェクトの中でいくつかのグループに分かれているのならば，リポジトリ内でのメンバー登録ではグループのメンバーだけにすること.

補足 2 各グループメンバーが作業を行う際に区別する必要があるため，ブランチを作ることで自分の作業を区別することができる（詳しくは「3.9.6GitHub Desktop の使い方」内の「ブランチの作成手順」を参照）.

（※文責: 小田涼平）

GitHub Desktop のインストール手順

主に次のサイトを参考に行った [16].

1. GitHub Desktop のページ（`https://desktop.github.com/`）を開く.
2. 画面上の「Download for Windows（64bit）」ボタンをクリックする (Windows, Mac, Linux の内から対応する物を選ぶ).

（※文責: 宮崎隼）

GitHub Desktop のサインイン手順

1. ダウンロードしたインストーラー（今回の場合「GitHubDesktopSetup-x64.exe」）を開く.
2. 開くと「Welcome to GitHub Desktop」の画面になり，「Sign in to GitHub.com」をクリックする.
3. GitHub アカウントのユーザー名かメールアドレスを「Username or email address」に入力する.
4. GitHub アカウントのパスワードを「passward」に入力して，「Sign in」をクリックする.
5.「Authorize GitHub Desktop」と表示され, GitHub Desktop を承認するために「Authorize desktop」をクリックする.
6.「GitHunDesktop.exe を開きますか？」と表示されたら，「GitHubDesktop.exe を開く」をクリックする.
7.「Configure Git」に遷移され，すべてデフォルトのまま「Finish」をクリックする.
8.「Let’s get started!」に遷移されたら，サインインまで完了したことになる.

（※文責: 宮崎隼）

3.9.6 GitHub Desktop の使い方

主に次のサイトを参考に行った [15][17].

GitHub からリポジトリのクローン方法

1.「GitHub Desktop」を起動する.
2. 左上の「File」をクリックする.
3.「Clone repository」をクリックする.

4.「Clone a repository」画面が表示され，「GitHub.com」の中にある自分のグループのリポジトリを選択する．
5.「Local path」は自分の分かりやすいものに設定する．
6.「Clone」をクリックする．

URL を用いたクローン方法

1.「Clone a repository」画面が表示されたら，「URL」を選択する．
2. Web 上の GitHub を開く．
3. クローンしたいリポジトリを開く．
4. 緑色の「<> Code」をクリックする．
5.「HTTPS」を選択し，表示されている URL をコピーする．
6. GitHub Desktop に戻る．
7.「URL or username/repository」にコピーした URL をペーストする．
8.「Local path」は自分の分かりやすいものに設定する．
9.「Clone」をクリックする．

補足 リポジトリを作ってからすぐにクローンを行った場合，「GitHub.com」内に作成したリポジトリが反映されていないことがあるため，その際は「URL を用いたクローン方法」を行うこと．

（※文責: 宮崎隼）

ブランチの作成手順

1. GitHub Desktop を起動する．
2.「Current repository」からブランチを作成したいリポジトリを選択する．
3. 画面上部の「Current repository」の横にある「Current branch」をクリックする．
4.「New branch」をクリックする．
5.「Name」に自分でつけたい名前を付ける．
6.「Create branch」をクリックする．
7. すると「Current branch」の横が「Publish branch」に変わるため，「Publish branch」をクリックする．
8. 画面下にある「Open the repository page on GitHub in your browser」内にある「View on GitHub」をクリックする．
9. ブラウザに GitHub が表示され，画面右にあるブランチのプルダウンを開き，上記で作成したブランチが表示されていることを確認する．

補足 1「6.「Create branch」をクリック」しても GitHub Desktop 上でしか反映されていないため，「Publish branch」を忘れずクリックすること．
補足 2 反映されているかどうかを確認するのなら，「Current repository」を右クリックし，「View on GitHub」を選択すること．
補足 3 ブランチの名前はグループで活動する場合，自分の学籍番号など自分だと分かりやすいものにすること．

（※文責: 宮崎隼）

コミットの方法

コミットはバグが無くなるなどして，作業がひと段落ついた際に行うもの．

1. GitHub Desktop を起動する．
2. 左にある「Change」を選択しファイルの追加や変更などを確認する．
3.「Summary」欄に簡単な更新内容を，「Description」に更新についての説明を入力する．
4.「Commit to ○○」をクリックする．

コミットを間違えた場合

左下にある「Committed ○○ ago 」の「Undo」をクリックする．

補足 コミットしたかどうかを確認したい場合，「History」を選択するとコミットの有無の履歴を確認することができる．

（※文責: 宮崎隼）

プッシュの方法

ある程度コミットがまとまってきた際に，GitHub にプッシュして同期するもの．

1. GitHub Desktop を起動する．
2. 右上にある「Push origin」をクリックする．

（※文責: 宮崎隼）

3.9.7 Unity プロジェクトを GitHub Desktop で管理する方法

主に次のサイトを参考に行った [18].

グループの一人が行う

1. GitHub Desktop を起動する．
2. 左上の「File」をクリックする．
3.「New repository」をクリックする．
4.「Name」にリポジトリの名前を書く．
5.「Description」に簡潔な説明を入力する．
6.「Local path」は自分の分かりやすいものにしておく．
7.「Initialize this repository with a README」をチェックする．
8.「Git ignore」は「Unity」を選択する．
9.「License」は「None」のままにする．
10.「Create repository」をクリックする．
11. 右上の「Publish repository」をクリックする．
12.「GitHub.com」を選択し，「Name」にファイルの名前を，「Description」は上記の説明を再度書く．

13.「Organization」は「3.9.4 GitHub の利用手順」内の「プロジェクト全体の共有設定」で作ったものを選択する．
14.「Publish repository」をクリックする．
15. Web 上の GitHub に上記のリポジトリがアップされていることを確認する．

各メンバーで行う

1. 上記のリポジトリをクローンする（「3.9.6 GitHub Desktop の使い方」内の「GitHub からリポジトリのクローン方法」を参照）．
2. クローンしたリポジトリに各自のブランチを作成する（「3.9.6 GitHub Desktop の使い方」内の「ブランチの作成手順」を参照）．
3. Unity Hub を起動する．
4.「New project」をクリックする．
5. 2D，3D など各々作りたい形式のものを選択する．
6.「Project name」を自分の好きなものにする．
7.「Location」はリポジトリをクローンする際に「Local path」で設定したものにする．
8. Unity プロジェクトが出来上がったら，「Local path」で設定したフォルダを開く．
9. Unity プロジェクトのフォルダなどがある中で「.gitignore」ファイルをコピーする．
10. Unity プロジェクトのフォルダを開き，「.gitignore」ファイルをペーストする．
11. GitHub Desktop でコミットする．
12. GitHub Desktop でプッシュする．

これにより Unity で作成，変更したものは GitHub Desktop でコミット，プッシュすることによって GitHub 上に反映することができる．

補足 1「Local path」で指定するフォルダは半角英数字にすることでエラーを回避することがある (自分の名前などの漢字が入っている場合にエラーが起きることがある)．

補足 2 Unity プロジェクトのフォルダにペーストする「.gitignore」は GitHub Desktop からリポジトリを作成する際に自動で生成したものであること．

補足 3「.gitignore」を Unity プロジェクトのフォルダ内に位置することでコミットすることができるようになる（しなければコミットの内容が「300 +」と表示され，コミットする容量が多くなってしまう）．

補足 4「各メンバーで行う」内の「2．クローンしたリポジトリに各自のブランチを作成する（「3.9.6 GitHub Desktop の使い方」内の「ブランチの作成手順」を参照）.」は，ブランチを作成していない場合のものである．ブランチが必要ないなどの場合は飛ばしてもよい．

補足 5 もしすでに Unity のプロジェクトを作っており，そのプロジェクトを GitHub に送りたい場合，Unity プロジェクトのファイルをリポジトリをクローンする際に「Local path」で設定した位置に移動させることで管理できるようになる．

（※文責: 宮崎隼）

第 4 章　アプリケーション「HEARTBEAT AIRLINE」の制作

この章では後期に活動したこと，及びその成果物について述べる．

（※文責: 宮崎隼）

4.1　目的・目標

前期では，ARG と SWG の二つに分け，それぞれの目的を達成するために動いた．しかし後期では，当初の予定通り二つのグループをまとめ，一つのアプリケーション開発を行う．そのため本プロジェクトの後期は，ユーザーの興味・関心を引き出した運動をはじめとする活動が楽しくなるようサポートするアプリケーションを開発することを目的とする．その目的を達成するために，前期の二つのグループ活動を生かし，スマートウォッチや活動量計からリアルタイムで取得した心拍数を用いたアプリケーション開発を目標とする．

1. 心拍数を扱ったアプリケーション開発
2. 実験とアプリケーションのデザイン考案

上記の二つがアプリケーション開発の具体的な内容であり，「1. 心拍数を扱ったアプリケーション開発」を ARG が前期取得した Unity の知識を利用して開発し，「2. 実験とアプリケーションのデザイン考案」を SWG が担当することにする．

（※文責: 宮崎隼）

4.2　テーマの決定

中間発表をふまえて，本プロジェクトは後期より今回開発するアプリケーションのテーマを決定した．まず，本プロジェクトで心拍数をリアルタイムで活用するアプリケーションのアイデアを出し合った．そこで出たアイデアの抜粋を下記に示す．

1. ダーツやジェンガなど既存のゲームに心拍数の要素を入れたアプリケーション
2. 心拍数の変化に応じてプレイヤーが驚く仕掛けが起こるアプリケーション
3. 運動中に心拍数を利用したアプリケーション

これらのアイデアの中で，本プロジェクトは運動中に心拍数を利用したアプリケーションというアイデアに注目した．運動中と言ってもサッカーのように常にプレイに集中しなければならない激しいスポーツでは，プレイ中に心拍数を利用したアプリケーションを使用することは難しい．ランニングのような体の動き以外に集中することがない運動であれば，その運動中に心拍数を利用したアプリケーションを使用することは激しいスポーツの場合に比べて明らかに容易である．それをふまえて，ランニングマシンで運動をしているときに使用できるアプリケーションを開発するというア

イデアが出た．ランニングマシンで運動をしているときは走ること以外に集中することがなく暇になりやすい．ランニング中は開始するときに心拍数が上昇し，終了したあとは心拍数が下がっていくと考えられる．つまり，開始から終了までで心拍数の変化が起こりやすい．ランニングマシン中に使用できる心拍数を利用したアプリケーションは，心拍数のリアルタイムでの活用ができると本プロジェクトは考えた．そこで，ランニングの開始から終了までの想定される心拍数の変動を予測した．開始時の心拍数は一気に上昇し，走行中は一定の心拍数を保ち，終了後はゆっくりと心拍数が下がると考えた．この動きは飛行機のように見えると本プロジェクトは考えた．飛行機は離陸するときには機体を上昇させ，飛行中は一定の高度を保ち，着陸するときには少しずつ高度を下げていく．この動きは心拍数の動きと似ていると考えられる．飛行機をテーマとしたアプリケーションを開発することに決定した．

（※文責: 尾崎篤史）

4.3 実験

4.3.1 目的

飛行機は出発地から離陸するときには高度を一気に上げ，目的地まで飛行しているときは一定の高度を保ち続け，目的地に着陸時はゆっくりと高度を下げていく．本プロジェクトは，運動時の心拍数の動きは飛行機のように運動開始時は上昇し，運動を続けていくと一定の心拍数を保ち，運動終了時はゆっくりと下がると考えている．しかし，それは仮定であり，実際には運動時の心拍数がこのように変動するとは限らない．よって，実際の運動時の心拍数が想定しているように変動するかを確かめる必要がある．また，運動終了後にすぐ運動をやめたときと，運動終了後に少しずつ運動を軽くしてやめたときに，この 2 つに心拍数の下がり方に差があるのかを確かめる必要がある．この実験では運動時の心拍数が想定しているものと同じように変動するかどうか，運動終了後のやめ方で心拍数の下がり方に差があるのかの 2 つを検証することを目的とする．

（※文責: 尾崎篤史）

4.3.2 方法

被験者

平均年齢 21 歳の男子大学生 4 人

場所・日時

公立はこだて未来大学 1 階トレーニングルーム．2022 年 11 月 4 日の 15 時 30 分から 17 時 30 分に行った．

装置

ランニングマシン，VIVOSMART4（以下スマートウォッチとする），パソコン，実験用アプリケーション，ANT+ ドングル

実験用アプリケーション

この実験を行うにあたって，心拍数を記録するためのアプリケーションを開発した．機能としては心拍数を記録する，リアルタイムでグラフ化するの 2 つの必要最低限な機能にとどめ，後に開発するアプリケーションにも流用できるようにした．スマートウォッチから心拍数をデータとして取得し，それをリアルタイムでグラフとして書き起こして表示する．また，右にあるボタンを押すと現在の心拍数が毎秒 4 回記録される．もう一度ボタンを押すと心拍数の記録が終了する．記録した心拍数のデータは csv ファイルとして出力される．csv ファイルには計測した時間とそのときの心拍数が記録される．

手続き

被験者は腕にスマートウォッチを装着し，ランニングマシンを使用して走行する運動行った．回数は 1 人 2 セット行うとした．はじめに，被験者の腕にスマートウォッチを装着させて正しく心拍数が計測されているかをスマートウォッチ，実験用アプリケーションの 2 つともを確認した．次に，被験者はランニングマシンに乗ってスタートボタンを押し，そのままランニングする運動を始めてもらった．運動時の走行速度は実験で使用したランニングマシンで最高速度だった時速 8 キロメートルとした．スタートボタンを押し，ランニングマシンが動き出した瞬間に実験用アプリケーションでボタンを押して心拍数の記録を始めるとした．運動時間は 2 セットとも 10 分間とした．1 セット目は 10 分間のランニングの後すぐに走行をやめさせ，ランニングマシンから降りてもらった．その後，実験用アプリケーションで表示している心拍数のグラフを目視して確認した．2 セット目は 10 分間のランニングの後に時速 5 キロメートルの走行を 30 秒，その後に時速 3 キロメートルの走行を 30 秒行わせた後に走行をやめさせた．その後，1 セット目と同じように実験用アプリケーションで表示している心拍数のグラフを目視して確認した．運動終了後は 1 セット目も 2 セット目も開始時の心拍数に戻ったとき，もしくは 1 分間経過しても心拍数の変化が見られなかったときに実験用アプリケーションでボタンを押して心拍数の記録を終了するとした．被験者に負担をかけさせないよう，1 人目の 1 セット目の後に 2 人目の 1 セット目，4 人目の 1 セット目の後に 1 人目の 2 セット目のように 4 人の被験者をローテーションさせて実験を行った．

（※文責: 尾崎篤史）

4.3.3 結果

実験で記録したデータをもとに心拍数の変化の様子，被験者ごとの心拍数データの違いを分析した．最大心拍数，最小心拍数のどちらもが被験者によってかなり異なっていることが分かった．運動開始から 2 分の傾きを見ると，4 人目を除いてほぼ同じような数値が確認できた．運動継続時に心拍数がある程度保たれていることがどの被験者とも見られた．しかし，10 分の運動終了後の傾きを見ると，運動開始から 2 分の傾きよりも個人差が大きいことが見られた．また，1 セット目と 2 セット目で運動後に急にやめさせるか，ゆっくりと運動をやめさせるかで運動のやめ方を変更したときの心拍数の下がり方に違いがあるのかを検証した．被験者全員の心拍数の変化を比べても 1 セット目と 2 セット目の間で傾きが緩くなっていたことが見られた．

被験者ごとの心拍数データと，特に顕著に心拍数の変化が見られた 2 人目と 4 人目の心拍数のグラフを下記に示す．

表 4.1　1 セット目の各被験者ごとの心拍数データ

	最大/最小心拍数	開始 2 分の傾き	10 分終了後の傾き
1 人目	トラブルによりデータなし		
2 人目	145/78	0.4851	-0.3162
3 人目	169/84	0.5056	-0.2127
4 人目	148/81	0.3611	-0.2151

表 4.2　2 セット目の各被験者ごとの心拍数データ

	最大/最小心拍数	開始 2 分の傾き	10 分終了後の傾き
1 人目	168/98	0.4526	-0.1968
2 人目	141/68	0.4623	-0.2020
3 人目	173/101	0.4501	-0.1730
4 人目	124/84	0.1815	-0.1587

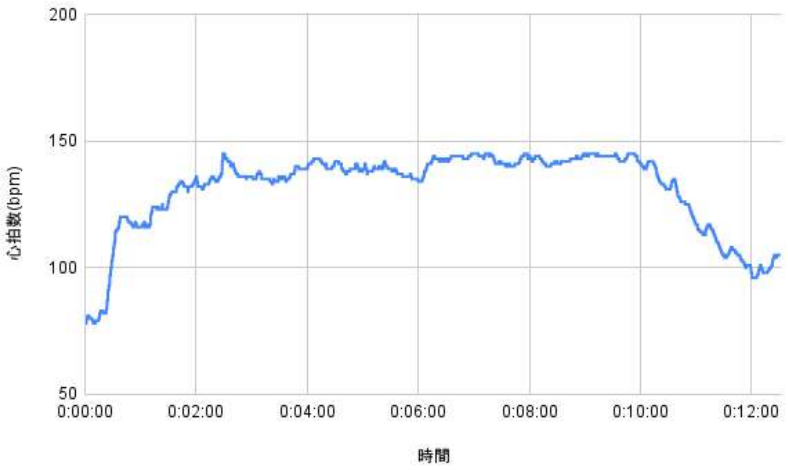

図 4.1　2 人目の 1 セット目での心拍数

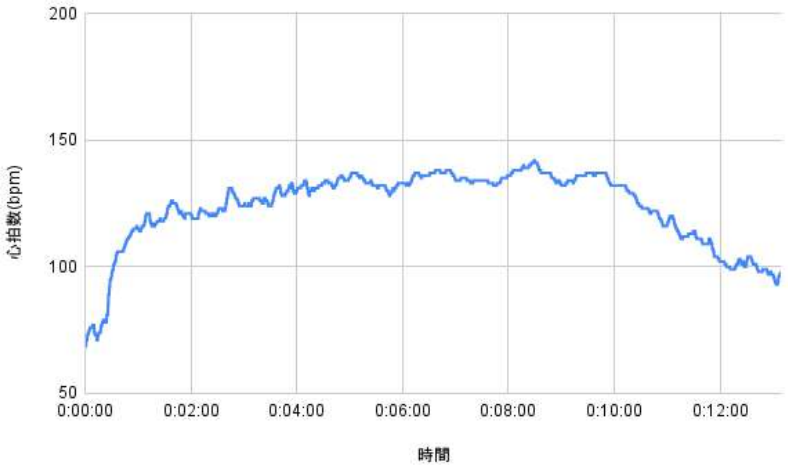

図 4.2　2 人目の 2 セット目での心拍数

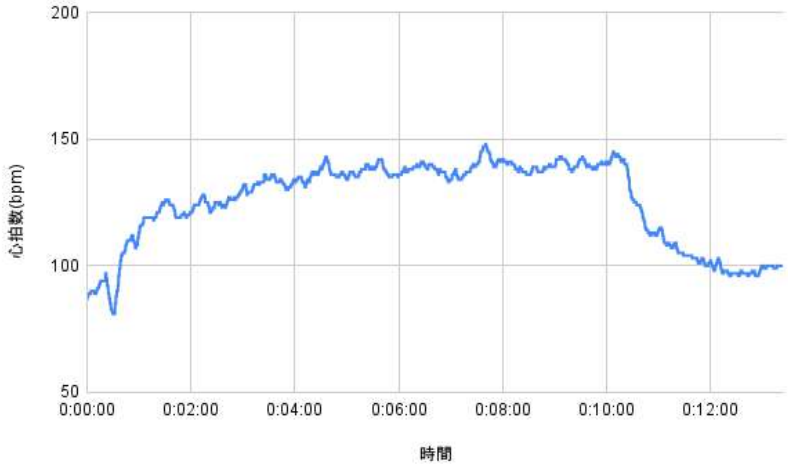

図 4.3　4 人目の 1 セット目での心拍数

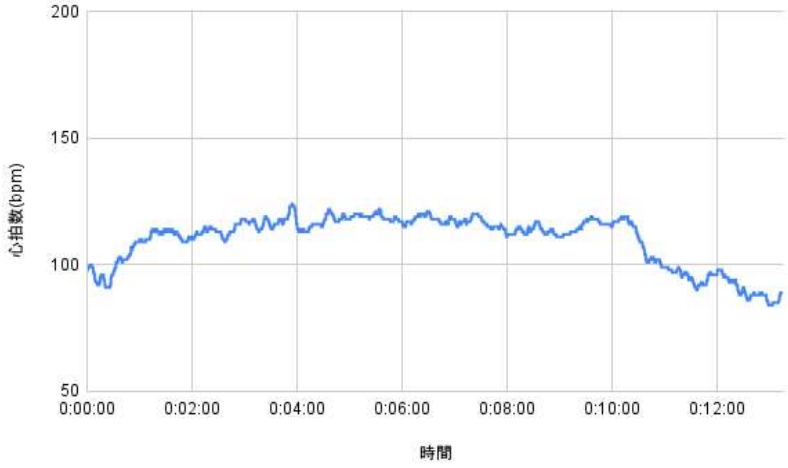

図 4.4　4 人目の 2 セット目での心拍数

（※文責: 尾崎篤史）

4.3.4　考察

運動時の心拍数の動きは飛行機のように運動開始時は上昇し，運動を続けていくと一定の心拍数を保ち，運動終了時はゆっくりと下がるのかを検証した．その結果，運動開始時の心拍数の上昇と，運動継続時の心拍数の保持はどの被験者とも同じように見られた．しかし，運動終了時の心拍数の下がり方は被験者によって異なることが分かった．運動終了後の心拍数の下がり方はその人の体力があるほど下がりやすい．これは被験者の体力の違いが原因であると考えられる．運動時の心拍数の動きは飛行機のように運動開始時は上昇し，運動を続けていくと一定の心拍数を保ち，運動終了時はゆっくりと下がるという想定は半分実証された．ランニング中に運動中の心拍数の上昇と保持は有効に利用できそうだが，運動終了後の心拍数の下降を利用するのは個人の差が大きいので難しいと本プロジェクトでは考えた．ランニングマシン中に使用できる心拍数を利用したアプリケーションには，飛行機の離陸と飛行までを実装することとした．また，運動終了後にすぐ運動をやめたときと，運動終了後に少しずつ運動を軽くしてやめたときに，この 2 つに心拍数の下がり方

に差があるのかをこの実験で検証し，それが実証された．しかし，ランニング中にプレイできる心拍数を利用したアプリケーションにこの性質を落とし込むのは難しいと本プロジェクトは考えた．

（※文責: 尾崎篤史）

4.4 成果物

本プロジェクトでは，これらの実験から運動中における心拍数の変動が航空機の軌道に類似していると考えた．また，前期 ARG の成果物を活用した Unity 開発環境を用いて成果物の目標を Unity を用いたゲームの開発とした．これらのことから自らの心拍数を用いて航空機の高度を操作する Windows 向け 2D ゲームアプリを開発することにした．開発したゲームの名前は「HEARTBEAT AIRLINE」とした．このゲームは主に 2 つのアプリケーションから構成されており，前期 ARG の成果物を活用した心拍数受信システムと，Unity を用いて開発したゲームアプリ本体である．各アプリケーションの主な仕様は以下表 4.3 の通りである．

表 4.3　成果物の主な仕様一覧

仕様	心拍数受信システム	ゲームアプリ
OS	Windows 10 Home 64bit	
RAM	8GB 以上	
画面解像度	-	FHD(1920 × 1080)
開発プラットフォーム	Java Script	Unity 2021.3.3f1
Node.js バージョン	16.17.1	-

それぞれのアプリが格納されているフォルダの構造は以下のようになる．なお，記載を省略している箇所がある．

- 心拍数受信システム

```
hertbeat-js
└── hertbeat-js
    │   .gitignore
    │   LICENSE
    │   log.txt
    │   main.js
    │   mainjs.bat
    │   package-lock.json
    │   package.json
    │   README.md
    │
    └── node_modules
        └...
```

- ゲームアプリ

```
build_test_1207
└── build_test_1207
        |       Project_ANT+.exe
        |       UnityCrashHandler64.exe
        |       UnityPlayer.dll
        |
        ├── MonoBleedingEdge
        |       └──...
        ├── Project_ANT+_BurstDebugInformation_DoNotShip
        |       └──...
        └── Project_ANT+_Data
                ├── Managed
                |       └──...
                ├── Plugins
                |       └──...
                ├── Resources
                |       └──...
                ├── StreamingAssets
                |       config.json
                |       HRMeasure.csv
                |       record.json
                |       UnityServicesProjectConfiguration.json
                └...
```

（※文責: 網干界）

4.4.1 心拍数受信システム

スマートウォッチからパソコンに心拍数を取り込むシステムは，第 2 章の前期 SWG が使用したものを改良して使用した．前期 SWG では ANT+ のドングル及びドライバを介して受け取った情報を JavaScript で処理し，HTML で可視化するシステムであった．一方で後期に本プロジェクトが使用する開発プラットフォームである Unity においては，外部から情報を受け取るために C#での制御が必要である．しかし，JavaScript から C#へ直接情報を受け渡しするための手段は見つからなかった．従って，JavaScript から C#に情報をリアルタイムで変換または送信する仕組みが必要であった．本プロジェクトではこの解決方法として，JavaScript と C#の間に node.js サーバーを中継させ，各ノード間を WebSocket 及び WebSocket-sharp と呼ばれる通信プロトコルを用いることにより，C#で JavaScript からの情報を受信できる心拍数受信システムを作成した．このシステムを用いることにより，JavaScript で取得した心拍数を Unity 内の C#スクリプトから受信できるようになった．以下図 4.5 にその模式図を示した．

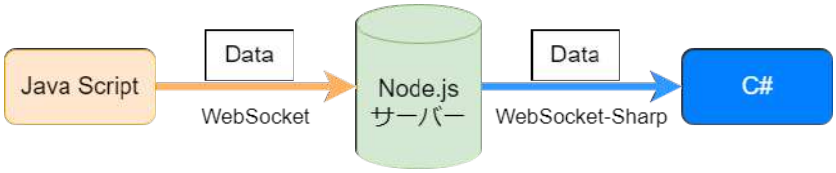

図 4.5　心拍数受信システムの模式図

- 使用方法
  1. 第 2 章の前期 SWG のシステムが正常に動作してる状態を想定する．
  2. Node.js サーバーの構築をする．詳細は後述する使用した技術及び付録を参照.
  3. 成果物に含まれる mainjs.bat を起動する．
  4. コマンドプロンプトが起動し，スマートウォッチのデバイス ID や心拍数，タイムスタンプなどが表示及び更新されていれば正常に動作している．

（※文責: 網干界）

4.4.2　ゲームアプリ

心拍数受信システムを用いて Node.js サーバーへ送信された心拍数を，Unity 内の C#スクリプトを用いて受信し，これをゲーム内におけるプレイヤーが操作可能な要素として実装し，ゲームとして開発を行った．

本ゲームはプレイヤーが，事前に設定した目標心拍数（HR_tgt）と，それに対する許容範囲（HR_width）内に，運動中の心拍数をどれだけ長い時間維持することが出来るかを競うものである．心拍数の変化は，飛行機を模したゲーム内オブジェクトの垂直方向の位置（高度）によって表される．心拍数をどれだけ許容範囲内に維持できているかは，許容範囲内をセーフゾーン，許容範囲外をデッドゾーンと呼称し，飛行機がセーフゾーン内に入っているか否かといった視覚的効果と，セーフゾーン内にいる秒数の合計として表されるスコアが加算されることにより判断できる．

ゲームのシーンは主に 3 つに分けられる．第 1 シーンはタイトル画面から通常時心拍数測定画面までである．第 2 シーンは飛行機の離陸から水平飛行に移るまでである．第 3 シーンは飛行機の高度と心拍数の変化が対応しているシーンである．

ゲームプレイまでの流れ

1. 付録 2.4 に従い，心拍数が取得できている状態を想定する．
2. 使用者は任意のスマートウォッチ（ANT+ 規格の通信を用いて心拍数を送信できるもの）を用意し，装着する．本プロジェクトでは garmin 社製のスマートウォッチ vivosmart4 を使用した．
3. ゲームを実行するパソコンにドングルを装着する．
4. スマートウォッチの設定を心拍数が送信できる設定に変更する．
5. 心拍数受信システム（mainjs.bat）を起動し，心拍数が正しく受信できていることを確認する．
6. ゲームアプリ（Project_ANT+.exe）を起動する．

ゲームのプレイ

1. ゲームを起動するとまずタイトル画面が表示される．このシーンでは中央に配置されて

いる「START」ボタンをクリックすることで次のシーン（通常時心拍数測定画面）へ遷移することができる．

2. 通常時心拍数測定画面では 30 秒間の平均心拍数を測定することができる．なお，このシーンは「skip」ボタンをクリックすることでスキップすることができる．
3. 飛行機の離陸シーンではプレイヤーの心拍数が許容範囲内に入るまで飛行機が上昇する．
4. プレイヤーの心拍数が許容範囲内に入ると画面上部から雲が現れた後，心拍数と飛行機の高度が対応するシーンへと変異する．

図 4.6　ゲームアプリのタイトル画面

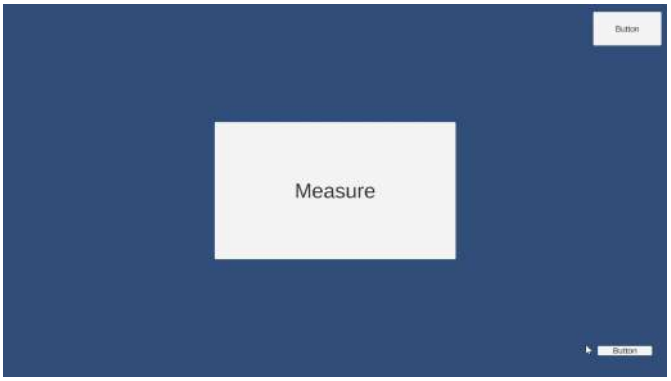

図 4.7　ゲームアプリの通常時心拍数測定画面

図 4.8　ゲームアプリの通常時心拍数測定画面をスキップするウィンドウ

（※文責: 網干界）

4.4.3 使用した技術

1. ANT/ANT+ 受信

概要

ANT とは，ある機器から別の機器に情報を無線で送信するための超低消費電力（ULP）無線プロトコルである [19]．ANT+ とはデバイスプロファイル [20] によって定義される，一貫した方法でネットワーク上にデータを送信する ANT のネットワークである [21]．また，これらの通信には 2.4GHz 帯の周波数が用いられている．

本ゲームでの実装方法

本ゲームにおいては主に心拍数の取得に ANT/ANT+ の技術を用いた．4.4.1 で示した心拍数受信システムに用いられ，スマートウォッチと USB ドングルの間を接続している．この心拍数受信システムでは最大で 4 回/秒の心拍数の受信が可能である．

2. webscket と Node.js サーバーを用いた通信

概要

webscket とは，RFC6455[22] によって定義されている双方向通信プロトコルである．RFC6455 の概要を引用すると，WebSocket プロトコルは，制御された環境で信頼できないコードを実行しているクライアントと，そのコードからの通信にオプトインしているリモートホストとの間の双方向通信を可能にする，とある．つまり，任意の実行環境にあるクライアントと，このクライアントからの通信を許諾しているリモートホストとの間の通信を可能にするものである．websocket-sharp とは，C#向けに設計された webscket 通信ライブラリである [23][24].

本ゲームでの実装方法

本ゲームでは主に心拍数受信システムと Node.js サーバー間の通信に webscket を，Node.js サーバーと Unity 間の通信に webscket-sharp を用いた．心拍数受信システム（mainjs.bat）で受信した心拍数データを，webscket を用いて Node.js サーバーへと送信している．次に Unity 内の C#スクリプトからサーバーに通信を行い，心拍数データを受信している．このとき，心拍数データは float 型の変数として扱われている．

Unity への WebSocketSharp プラグインの導入方法 [25][26][27][28]

（a） WebSocketSharp の GitHub[23] から，リポジトリをクローンまたは zip ファイルをダウンロードする．本プロジェクトでは zip ファイルのダウンロードを行う方法を用いた．

（b） ダウンロードした zip ファイルを解凍（展開）する．

（c） 解凍（展開）したファイル内の websocket-sharp.sln を Visual Studio で開く．

（d） Visual Studio のソリューションエクスプローラーより，websocket-sharp をビルドする．このとき，.NET Framework の開発環境が導入されていない，または古い場合にダウンロードインストールが要求されるので指示に従い導入する．なお，対象のフレームワークは.NET Framework 3.5 とする．

（e） ビルドが完了すると websocket-sharp/bin/Release/websocket-sharp.dll が生成される．このとき，websocket-sharp/bin/Debug や websocket-sharp/bin/Debug/Ubuntu 内に websocket-sharp.dll が生成されることがあ

る．これを修正したい場合はビルド-構成マネージャーからアクティブソリューション構成と構成を Release に変更する．

(f) 生成された dll ファイルを，Unity プロジェクトの Assets/Plugins 内に入れる．このとき，Plugins フォルダがない場合は作成する．

3. ゲームシーンの取り扱い（SceneManager）

概要

Unity におけるシーン（Scene）とは，Unity のコンテンツを扱う場所であり，ゲームやアプリケーションの全部または一部を含むアセットである [29]．SceneManager は実行時のシーン管理を行うクラスである [30]．Unity における実行時のシーン管理をスクリプトから行うためのクラスである．シーンに関する情報や，シーンのイベントを管理することが出来る．

本ゲームでの実装方法

本プロジェクトで主に使用した機能は，シーンをロードする関数（SceneManager．LoadScene[31]），シーンをアンロードする関数（SceneManager．UnloadScene[32]）である．UnloadScene は，公式ドキュメントでは廃止されており，UnloadSceneAsync を使用することが推奨されている．本ゲームでは 5 つのシーンが用意されているが，実際に実装されているシーンは 4 つであり，機能しているものは 3 つである．それぞれのシーンについてまとめた表が以下表 4.4 である．

表 4.4　本ゲームで使用したシーンの一覧と概要

シーン名	用意	実装	機能	概要
00_start	YES	YES	YES	スタート画面
01_MeasureHR	YES	YES	YES	通常時心拍数の測定
02_main	YES	YES	YES	ゲームプレイ画面
03_result	YES	NO	NO	スコアや記録の表示（未実装）
04_option	YES	YES	NO	各種設定（機能なし）

4. ボタン（Button）

概要

ボタンは，テキストラベルからなり，クリックすることでポインタイベントやマウスイベントに応答することができる [33]．

本ゲームでの実装方法

本ゲームではユーザーは画面内のボタンのクリックによってイベントやシーンの操作を行うことができる．

スタート画面には中央にスタートボタンと，右上に設定ボタンがある．スタートボタンをクリックすると，次のシーンである通常時心拍数測定画面に遷移する．設定ボタンを押すと設定画面へと遷移する．

通常時心拍数測定画面には，中央に測定ボタン，右下にスキップボタン，右上に設定ボタンがある．測定ボタンを押すとボタンの色が変化し，30 秒間の心拍数測定が開始される．30 秒経過後に平均心拍数が表示され，ゲームプレイ画面への遷移ボタンへと変化する．スキップボタンは心拍数測定画面をスキップし，ゲームプレイ画面へ遷移す

ることができる．なお，確認のためのポップアップウィンドウが表示される．右下のスキップボタンを押せばスキップができ，右下のキャンセルボタンもしくは右上のバツボタンを押すとポップアップウィンドウは消える．設定ボタンの機能はスタート画面のものと同様である．

設定画面には元の画面に戻るためのボタンのみがある．

5. ゲームオブジェクトのスクリプト制御

概要

Unity 内の全てオブジェクトにはトランスフォーム（Transform）コンポーネントが存在し，各オブジェクトの位置（Position），回転（Roatation），スケール（Scale）を定めている [34]．これらの値は Unity 内のインスペクター上から，もしくはスクリプトを用いて変更することができる．スクリプトを用いる場合，位置は Transform．position，回転は Transform．rotation，スケールは Transform．localScale を用いて取得することができる．位置，スケールは x，y，z 軸の 3 次元ベクトル（Vector3）で，回転は 4 次元ベクトル（Quaternion）である．なお，回転の 4 次元ベクトルは Quaternion．Euler を用いて 3 次元ベクトルから変換することができる [35]．本ゲーム内では画面の中心が原点，つまり x 座標，y 座標共に 0 である．

本ゲームでの実装方法

本ゲームでは主にゲームのプレイ中であるシーン，02_main において，以下のような流れでスクリプトを用いた位置の取得と更新を行っている．

（a） シーン遷移直後の各オブジェクトの Transform は，インスペクター上から設定した初期位置である．

（b） 画面左下にある飛行機が画面中央下部まで画面右に向かって移動する．

（c） 飛行機が画面中央下部まで来ると飛行機が回転し，中央まで画面上方向に向かって上昇を始める．同時に背景の移動に縦方向の移動が加わる．これによって飛行機が上昇しているように見える．

（d） 目標心拍数-許容範囲（HR_tgt-HR_width）以上になると雲を模したオブジェクトが画面上部描画範囲外から画面下描画範囲外まで移動する．

（e） 雲が画面を覆う瞬間に，飛行機の回転を元に戻し，飛行機と心拍数の同期を開始をしている．

（f） これ以降は，飛行機の y 座標を受けとった心拍数から目標心拍数を引いた値を与え続けている．

6. 飛行機の軌跡（LineRenderer）

概要

LineRenderer は，3D 空間で自由に動く線を描画するために使用されるコンポーネントである [36].

本ゲームでの実装方法

本ゲームにおいては心拍数の履歴を飛行機の軌跡に見立てて表示するために，LineRenderer を使用した．LineRenderer に座標配列を与えるとその間に線を描画できる．これを用いて，一定間隔で新たな心拍数から得られる飛行機の座標の追加と配列の更新を行うことで飛行機の軌跡に見立てて表示している．

7. json ファイルからの設定数値の読み込み（JsonUtility）

概要

JsonUtility は，json データを操作するためのユーティリティ関数である [37].

本ゲームでの実装方法

本ゲームにおいては，目標心拍数や許容範囲などのユーザーが設定できる設定数値や，ハイスコアなどの記録などを json ファイルを用いて管理している．これらのファイルはゲームアプリファイル内の Project_ANT+_Data/StreamingAssets 内に保存されている．設定ファイルは config.json で，記録ファイルは record.json である．それぞれのファイル内の変数の一覧と概要は以下の表 4.5，表 4.6 の通りである．

表 4.5　本ゲームで使用した設定ファイル（config.json）の一覧と概要

変数名	概要
HR_tgt	目標心拍数 [bpm]
HR_width	許容範囲 [bpm]
rate_de	降下率 [bpm/s] 未使用
rate_as	上昇率 [bpm/s] 未使用
t_as	上昇時間 [s] 未使用
t_de	降下時間 [s] 未使用
t_run	水平飛行（走行）時間 [s]
spd	背景とグラフのスピード [pos.x/f]
camera_scale	カメラスケール 未使用

表 4.6　本ゲームで使用した記録ファイル（record．json）の一覧と概要

変数名	概要
high_score	ハイスコア
HR_data_00	直前プレイの心拍数データへのパス
HR_data_01	前回プレイの心拍数データへのパス
HR_data_02	2 回前プレイの心拍数データへのパス
HR_data_03	3 回前プレイの心拍数データへのパス
HR_data_04	4 回前プレイの心拍数データへのパス
HR_nrm	通常時心拍数

（※文責: 網干界）

4.5　デザインの検討

この章では，後期成果物「HEARTBEAT AIRLINE」のデザインにおいて検討した内容について述べる．デザインは iPad を利用して，無料アプリケーションである「ibisPaint X」で制作した．イラストの画像は E に記載している．

（※文責: 尾崎篤史）

4.5.1　背景について

背景の画像サイズは 3840 × 1080 である．これは，2 画面分の大きさである．本アプリケーションでは画像中央が画面右端に到達するタイミングで画像をループさせることで背景を動かすという手法をとっているため，このサイズとなっている．また，滑走路と空港により，ターミナルから飛行機が離陸するまでを表現した．空は上昇が終了し機体が上空に到達した段階で用いている．作業としては，「ibisPaint X」の真ん中から対称的に絵を描くことができる対称定規の機能を利用することで端と端での色等の違和感をなくすことを行った．

（※文責: 尾崎篤史）

4.5.2　飛行機・上空の物体について

本アプリケーションでは機体に関して，小型の個人用飛行機なども検討した．しかし，心拍数を飛行機雲として表現可能な点やユーザーによって目標心拍数まで上昇がゆるやかで時間がかかる場合に上昇時間が長くても違和感なくイメージできるという点から大型航空機を採用した。

上空に物体（気球・風船・帽子・UFO など）を配置することも検討した．しかし，用いなくても上空の表現が可能である点・衝突などのイベントがゲームに存在しない点・ユーザーによって目標心拍数に早く到達した場合演出として現れない点を考慮してデザインとして組み込まなかった．

（※文責: 尾崎篤史）

4.6　成果発表会

この章では後期に行った（2022，12，9）成果発表会について述べる．成果発表会は，これまでのプロジェクト学習の成果を学内外の関係者に発表し，評価していただくことを目的としたものであり，一昨年，去年新型コロナウイルスの影響もあり，オンライン形式での発表だった．今年は 3 年ぶりの対面形式での開催となった．本プロジェクトは主にスライドとポスターを用いて発表し，実際にランニングマシーンで走ることによって成果物のデモを行った．また発表会終了後，評価者に記入していただいた評価フォームをもとに，集計・解析・検討を行った．

（※文責: 丹野陽翔）

4.6.1　成果発表会に向けて

まず，ランニングマシーンでデモをする実演者は激しい運動になる事が予想されるため，心肺への負荷を考慮して，実演者は走る際にはマスクを外すように指示した．そして，実際にランニングマシーンでデモを行うにあたって実演者と聴衆との間に十分な距離を確保できるように，実演者はトレーニングルームで行い，聴衆者は体育館から窓越しにトレーニングルームを眺めながら発表を聞く方式で行うことにした．また，発表会当日に速やかにデモを行うための練習として，事前にトレーニングルームを使用したデモを行い入念な準備を行った．その際にはデモ動画の撮影も行った．

（※文責: 丹野陽翔）

4.6.2　ポスター・スライドの作成

成果発表会に使用したポスターは Adobe Illustrator で作成した．成果発表会に使用したスライドは，Google スライドで作成した．また，スライドの 1 ページ目にはデモ動画を挿入した．

（※文責: 丹野陽翔）

4.6.3　成果発表会当日

本プロジェクトでは前半（網干，丹野，熊坂，安澤，川上）と，後半（八木田，尾崎，森，宮崎，小田）の 5 人ずつに分かれて 15 分間の発表をそれぞれ 3 回ずつ行った．

（※文責: 丹野陽翔）

図 4.9　成果発表会の様子

4.6.4　評価シートの集計・解析・検討

評価フォームには，評価者の種別や，学籍番号または所属，評価者指名，発表技術の 10 段階評価，発表技術についてのコメント，発表内容の 10 段階評価，発表内容についてのコメントの 7 項目があった．

（※文責: 丹野陽翔）

4.6.5　評価シートの枚数

前半で回収した評価フォームは 23 枚，後半で回収した評価フォームは 21 枚の計 44 枚の評価フォームが集まった．評価者の内訳は，本大学の学生が 40 枚，本大学の教授が 4 枚であった．

（※文責: 丹野陽翔）

4.6.6　評価の平均点

表 4.7　評価の平均点

評価内容	前半	後半	前後半の総合	中央値
発表技術	8.09	8.29	8.18	8
発表内容	7.70	7.43	7.57	8

上記の表は発表技術の 10 段階評価と，発表内容の 10 段階評価について前半，後半，前後半の総合の 3 つに分けてそれぞれ平均点を算出した．また，平均点の算出は有効数字 3 桁とし，小数点第 4 位を四捨五入したものである．表から発表内容，評価技術共に平均点に大きな差はないことから，前後半における発表技術，発表内容についてはほぼ同じものとみなすことが出来る．

（※文責: 丹野陽翔）

4.6.7　コメントからの今後の改善点

集まった評価フォーム計 44 枚のうち，発表技術についてのコメントと，発表内容についてのコメントから分析，考察を行った．詳しいフィードバックの内容については，4.6.9 の発表技術についてと 4.6.10 の発表内容についてに記載している．

（※文責: 丹野陽翔）

発表技術について

4.6.9 から，発表の前後半を総じて，「聞き取りやすい」，「わかりやすい」などの，前向きなコメントを多く頂いた．本プロジェクトの発表は複数のプロジェクトが同一の空間に集まる体育館で発表していたこともあり，中間発表の時から「声の小ささ」が度々評価シートでも指摘されていた．そこで成果発表会では中間発表会の時よりもより大きな声で発表できるように意識，練習してきた．結果，中間発表の時ほど「声の小ささ」について指摘しているコメントは見受けられなかったことから，コメントを頂いたとおりに聴衆にとって聞き取りやすい発表を行えただろう．また，発表の際に用いたスライドについても「わかりやすい」，「見やすい」などのコメントを多く頂いた．要点を抑え，聴衆に見せる文章量を減らしたことが功を奏したといえるだろう．さらに，アプリのデモンストレーションを実際に走って行うことによって，画像だけでは伝わりにくい点をリアルタイムで見せることは非常に有効な方法であった．

（※文責: 丹野陽翔）

発表内容について

4.6.10 の発表内容から，心拍数とゲームの操作を連動させる面白い発想と，ゲームの拡張性についてコメントを多く頂いた．本プロジェクトではルームランナーで使用することを目的としたアプリを作成したが，他の運動や，運動以外での使用も出来そうとのコメントも頂いた．また，ユーザーに対してスコアなどをまとめたリザルト画面などを報酬としてフィードバックを行ったり，目標心拍数の設定の指標などの提示など，ユーザー視点での追加可能な要素の貴重なアドバイスを頂いた．しかしアプリを作成しようと思った背景の説明や，X-Reality との関連性についての質問があったので，本プロジェクトの発表は完全とは言い切れる内容ではなかった．

（※文責: 丹野陽翔）

4.6.8　成果発表の 5 段階評価

本グループの成果発表会全体での自己評価は，5 段階評価で 4 である．その理由は，簡潔でわかりやすいスライドとポスターを用いて説明した後，更にルームランナーを用いてアプリのデモを行うことによって，本プロジェクトがどの様なものを作成したのかは最低でも伝えることが出来た．また，実際に走ってデモを行うことは大きなインパクトを聴衆者に与え，聴衆者からは私たちの予想を超える好評が得られた．しかし「発表内容について」で先述した通り，本プロジェクトには本アプリを作成するに至った背景や，X-Reality との関連性について説明不足な点が多かった．これらのことから，発表の見せ方は良かったが，発表内容の構成に改善の余地があると判断して 5 段階評価の 4 とした．

（※文責: 丹野陽翔）

4.6.9　発表技術についてのフィードバック

発表技術についての評価点（10 点満点）は，平均値は 8.18 点，中央値は 8 点であった．また，コメントのうち高評価であったもの（9 点以上）を一部抜粋したものを，以下にまとめる．

- 実際にランニングマシンで紹介していることでわかりやすかった．
- 画像を用いながら要点を細かくまとめ，できるだけ短く説明していて理解しやすかった．
- 聞き取りやすい速度で話してくれて，発表時間にちょうど収まっていた．声は少し小さかったが，聞こえないほどではなかった．
- デモが面白かった．
- ポスターがシンプルで見やすい．また，実演も交えているのでイメージがつきやすい．
- 技術の説明もわかりやすく，身体を張ったデモンストレーションで良かったと思います．ちょっと機体の動きが想定と違ったようですが，どう動かくは伝わるプレゼンだったと思います．もう少し声が大きいと，なお良かったと思います．
- 聞いている側を向いている時間が長かった．
- 実際に心拍数をとって反映する段階までもっていっていたので素晴らしいと思った．
- 使っている機器のパッケージが横においてあり，実機のイメージがつきやすかった．

次に，コメントのうち低評価であったもの（7 点以下）を一部抜粋したものを，以下にまとめる．

- 声が小さく聞き取りづらい箇所が多かった.
- 3 人のうち二人が手もとのメモをみていた.
- 若干聞き取りにくい箇所があった.
- 他の発表の声に紛れて聞こえづらい時があった.

（※文責: 丹野陽翔）

4.6.10　発表内容についてのフィードバック

発表内容についての評価点（10 点満点）は，平均値は 7.57 点，中央値は 8 点であった．また，コメントのうち高評価であったもの（9 点以上）を一部抜粋したものを，以下にまとめる.

- 心拍数を用いたゲームというのは非常に面白い発想だと感じた.
- リアルタイムでデータを反映するのを, ゲームにしているためすごくわかりやすかった.
- 内容を流れに沿って説明しているので分かりやすかった.
- ゲーム感覚で運動できるは良い.
- 運動以外にも使えそう.
- 運動とゲームを紐付けると運動のモチベーション維持につながる.
- 詳細な技術仕様に言及していてとても良いと思いました.
- 拡張性も高いテーマだと思った.
- ゲーム要素を取り入れたシステムは面白いアイデア.
- 「リアルタイムで」というキャッチコピーに合わせて，その場で実演をしていてとてもわかりやすかった．心拍数の上下を飛行機と合わせている発想もおもしろかった.
- ゲームオーバーなどで，ユーザーに健康を促せるような表示があるとさらにグッと良くなると思いました.
- 今後のいろいろな発展を期待させる面白い内容でした．健康応用以外にも，いろいろありそうに感じました.
- 飛行機を用いるゲームならやはり一連の流れ（離陸から着陸まで）をやったほうがよいと思った.
- 体育館での発表なのを生かした，運動してデモを行うのが面白かった.
- 心拍数と飛行機の高度を紐付けるアイデアと直感的なフィードバックがいいと思いました．楽しく運動できるのではないかと思います.

次に，コメントのうち低評価であったもの（7 点以下）を一部抜粋したものを，以下にまとめる.

- 今回のアプリケーションを作ろうと思った背景が不明だった（理由やプロジェクト内でのプロセスなど）
- ゲームの UI 面とプレイヤーへの報酬提示が不足しているように感じた
- 個人的に斬新なゲームは健康面を考えると良いゲームだと思いますが，人を選ぶようなゲームだと思いました．スコアに応じて報酬が貰えるとかの機能とか付けたら良いと思いました.

（※文責: 丹野陽翔）

4.7 HEARTBEAT AIRLINE 仕様書

4.7.1 概要

本プロジェクトにおけるゲームアプリケーションは，ANT+ 規格のスマートウォッチを用いた心拍数の可視化及び心拍数を用いたゲームを目的とするアプリケーションである．本ゲームの名称は「HEARTBEAT AIRLINE」である．

1. 心拍数受信システム
2. ゲームアプリケーション

心拍数受信システムの使用方法やインストール手順は別途実験用アプリ導入・利用マニュアル (付録 F) を参照．本アプリケーションは，以下の 2 つのシステムから構成されている．これらのシステムの開発環境は以下表 4.8 の通りである．

表 4.8 HEARTBEAT AIRLINE の主な開発環境

仕様	心拍数受信システム	ゲームアプリ
OS	Windows 10 Home 64bit	
RAM	8GB 以上	
画面解像度	-	FHD(1920 × 1080)
開発プラットフォーム	Java Script	Unity 2021.3.3f1
Node.js バージョン	16.17.1	-

これらのシステムのビルド済みフォルダは，本プロジェクトの Google ドライブ共有フォルダ（以下共有フォルダ，または project2022 と記載）内の以下のフォルダにある．

1. 心拍数受信システム
   project2022/後期/開発/成果物/heartbeat-js.zip
2. ゲームアプリケーション
   project2022/後期/開発/成果物/build_test_1207.zip

また，ゲームアプリケーションの Unity プロジェクトは GitHub を用いて管理しており，本プロジェクトの GitHub（以下 GitHub2022 と記載）にある．

1. ゲームアプリケーションの Unity プロジェクトフォルダ
   `https://github.com/projecttsunagu2022/AR/tree/1020107_Aboshi/ANT-plus-app01`

（※文責: 熊坂賢人）

4.8 ディレクトリ構造

心拍数受信システム及びゲームアプリケーションのビルド済みファイルのディレクトリ構造は以下の通りである．なお，システムファイルやライブラリなどは記載を省略している．

- 心拍数受信システム (hertbeat-js)

```
hertbeat-js
└── hertbeat-js
    │   .gitignore
    │   LICENSE
    │   log.txt
    │   main.js
    │   mainjs.bat
    │   package-lock.json
    │   package.json
    │   README.md
    │
    └── node_modules
        └...
```

- ゲームアプリケーション

```
build_test_1207
└── build_test_1207
    │   Project_ANT+.exe
    │   UnityCrashHandler64.exe
    │   UnityPlayer.dll
    │
    ├── MonoBleedingEdge
    ├── Project_ANT+_BurstDebugInformation_DoNotShip
    └── Project_ANT+_Data
        ├── Managed
        ├── Plugins
        ├── Resources
        └── StreamingAssets
            config.json
            HRMeasure.csv
            record.json
            UnityServicesProjectConfiguration.json
```

また，GitHub で管理している Unity プロジェクトフォルダは，GitHub2022 内の以下のディレクトリ内にあり，そのディレクトリ構造は以下の通りである．システムファイルなどは記載を省略している．

- GitHub

GitHub2022/AR/tree/1020107_Aboshi/ANT-plus-app01

```
├── Assets
│       ├── Materials
│       ├── Plugins
│       ├── Resources
│       ├── Scenes
│       ├── Scripts
│       │       │   CSVReader.cs
│       │       │   JSONManager.cs
│       │       │   SceneChangeManager.cs
│       │       │
│       │       ├── 00
│       │       ├── 01
│       │       │   ButtonScript01.cs
│       │       │   HeartRate_01.cs
│       │       │
│       │       └── 02
│       │           BackgroundManager.cs
│       │           DeadZoneManager.cs
│       │           HeartRate.cs
│       │           LineTracker.cs
│       │           PlayerManager.cs
│       │           ScoreManager.cs
│       │
│       ├── StreamingAssets
│       ├── TextMesh Pro
│       └── Textures
│
│   170_20221206224603.png
│   airplane.png
│   airport.png
│   backgorund.png
│   background_blue.png
│   background_land.png
│   balloon.png
│   clowd.png
│   heart-st1.png
│   heart-st2.png
│   hot_air_balloon.png
│   over_spotter.png
│   triangle.png
│   UFO.png
```

```
|
├── Packages
└── ProjectSettings
```

（※文責: 熊坂賢人）

4.8.1 プログラム構成

本アプリケーションで主に使用しているスクリプトは Assets/Scripts 内にあり，各スクリプトの概要を以下に示す.

表 4.9 HEARTBEAT AIRLINE の主な仕様

スクリプト名	フォルダ	概要
CSVReader.cs	Assets/Scripts	未使用
JSONManager.cs	Assets/Scripts	設定ファイルの読み込み 記録ファイルへの書き込み関数定義
SceneChangeManager.cs	Assets/Scripts	シーン管理 クリックされたボタンに応じたシーン読み込み
ButtonScript01.cs	Assets/Scripts/01	通常時心拍数測定ボタン
HeartRate_01.cs BackgroundManager.cs	Assets/Scripts/01 Assets/Scripts/02	心拍数取得 プレイ画面の背景移動
DeadZoneManager.cs	Assets/Scripts/02	デッドゾーン範囲の描画
HeartRate.cs	Assets/Scripts/02	心拍数取得 自機（飛行機）の座標と心拍数を同期 心拍数の csv 出力
LineTracker.cs PlayerManager.cs	Assets/Scripts/02 Assets/Scripts/02	飛行機の軌跡の描画 自機及びオブジェクトの統合管理
ScoreManager.cs	Assets/Scripts/02	スコアの管理

各スクリプトの詳細を以下に示す.

1. CSVReader.cs
   未使用
2. JSONManager.cs
   このスクリプトでは外部ファイルで管理している，設定ファイル（config.json）と記録ファイル（record.json）の読み込みと，記録ファイルへの書き込み関数の定義を行っている.
   各設定ファイル内の変数に対応した変数を持つクラスをそれぞれ作成し，読み込んだ設定ファイルを対応するクラスに変換し，外部参照できるように新たに用意した変数に格納する.
   WriteHighScore 関数は，読み込んだ記録ファイル内の high_score を，入力された high_score に更新し，記録ファイルに書き出す関数である.
   WriteHRnrm 関数は，WriteHighScore 関数の機能における high_score を，HR_nrm に

変更したものである．

表 4.10　JSONManager.cs の変数一覧

変数名	型	スコープ	説明
path_config	string	Global	設定ファイル（config.json）への相対パス
path_record	string	Global	記録ファイル（record.json）への相対パス
config_str	string	Global	設定ファイル内のテキストをすべて読み取った文字列
config	InputConfig	Global	JsonUtility.FromJson を用いて，config_str を json ファイルとして各変数の読み込み
HR_tgt	int	Global	json ファイルから読み込んだ変数を外部参照できるように格納する変数
HR_width, rate_de, rate_as, t_as, t_de, t_run, spd, camera_scale	float	Global	同上
record_str	string	Global	記録ファイル内のテキストをすべて読み取った文字列
record	Record	Global	JsonUtility.FromJson を用いて，record_str を json ファイルとして各変数の読み込み
high_score, HR_nrm	int	Global	json ファイルから読み込んだ変数を外部参照できるように格納する変数
HR_data_00, HR_data_01, HR_data_02, HR_data_03, HR_data_04	string	Global	同上
writerecord	string	Local	記録ファイルに書き込みをするために文字列へ変換するための変数

3. SceneChangeManager.cs

   このスクリプトでは，全体的なシーンの管理を行っている．

   Awake 関数は，最初に呼び出され，最初のシーンである 00_start をロードし，他のシーンをアンロードしている．

   OnClick 関数は，対応するボタンがクリックされた際に呼び出される関数で，クリックされたボタンの名前に応じて特定のシーンをロードしてる．

4. ButtonScript01.cs

   このスクリプトでは，心拍数測定画面（01_MeasureHR）内の，通常時心拍数測定ボタン

表 4.11　SceneChangeManager.cs の変数一覧

変数名	型	スコープ	説明
btn	Button	Global	クリックされたボタンを識別
SceneNumber	byte	Global	シーンを識別するための ID

の挙動を管理している．

Awake 関数では，毎秒フレーム数を 60fps に設定している．

Start 関数では，状態の初期化として不要な Canvas を非表示にして，HR_nrm_sum を 0 にしている．

OnClick 関数では，シーン内にあるボタンが押された際に，どのボタンが押されたかを識別し，各ボタンに対応する挙動や関数を呼び出している．

MeasureHRCoroutine 関数は，通常時心拍数測定に用いるコルーチンである．測定開始ボタンがクリックされると，30 秒間 1 秒ごとに心拍数を csv に出力し保存する．30 秒経過後，保存した csv ファイルを読み込み，記録されている心拍数の平均値を算出し，表示する．

表 4.12　ButtonScript01.cs の変数一覧

変数名	型	スコープ	説明
btn	Button	Global	クリックされたボタンを識別
skipCanvas	GameObject	Global	スキップ確認ウィンドウの Canvas オブジェクト
canvasPlayButton	GameObject	Global	スキップ確認ウィンドウ内の play ボタン
measureing	GameObject	Global	通常時心拍数測定中の Canvas オブジェクト
slider	GameObject	Global	通常時心拍数測定中の進行度確認グラフ
sliderFill	GameObject	Global	進行度確認グラフの可変部
heartrate	HeartRate_01	Global	心拍数取得のためのスクリプトオブジェクト
jsonmanager	JSONManager	Global	外部ファイル読み込み
path_csv	string	Global	通常時心拍数の計算のための一時保存ファイルのパス
HR_nrm_sum	int	Global	通常時心拍数の計算のための一時格納変数 1 秒ごとに 30 秒間計測した心拍数の合計
HR_nrm_ave	int	Global	通常時心拍数の計算のための一時格納変数 HR_nrm_sum/30
HR	float	Global	心拍数
btn_txt	TextMeshProUGUI	Global	心拍数測定ボタンのテキスト
HR_txt	TextMeshProUGUI	Global	心拍数表示用のテキスト
sw	StreamWriter	Global	csv 出力のための StreamWriter オブジェクト
buttonCount	byte	Global	ボタンが押された回数を記録

5. HeartRate_01.cs

   このスクリプトでは，心拍数測定画面（01_MeasureHR）で使用する心拍数をサーバーから取得する．詳細は後述する HeartRate.cs と同等であるため，そちらを参照.

6. BackgroundManager.cs

   このスクリプトでは，ゲームプレイ画面（02_main）内の背景の挙動を管理している．背景オブジェクトの挙動は PlayerManager.cs 内の status の状態によって変化している．背景の移動速度は設定ファイル（config.json）の spd を参照している．

表 4.13　BackgroundManager.cs の変数一覧

変数名	型	スコープ	説明
jsonmanager	JSONManager	Global	外部ファイル読み込み
playermanager	PlayerManager	Global	PlayerManager.cs の変数を参照するためのオブジェクト
Background	GameObject	Global	背景オブジェクト（空中）
Background_land	GameObject	Global	背景オブジェクト（地上）
AirPort	GameObject	Global	背景オブジェクト（空港）
BackgroundPosInit	Vector3	Global	背景オブジェクト（空中）の初期位置の Vector3
BackgroundLandPosInit	Vector3	Global	背景オブジェクト（地上）の初期位置の Vector3
spd	float	Global	背景移動のスピード 外部の設定ファイル（config）の spd 参照
status	int	Global	プレイヤーの状態 PlayerManager.cs 内の status 参照

7. DeadZoneManager.cs

   このスクリプトでは，ゲームプレイ中のデッドゾーン範囲を示すオブジェクトの描画を管理している．

表 4.14　DeadZoneManager.cs の変数一覧

変数名	型	スコープ	説明
DZ_Up	GameObject	Global	上部デッドゾーン範囲
DZ_Lw	GameObject	Global	下部デッドゾーン範囲
HR_tgt	float	Global	設定ファイルの HR_tgt 参照
HR_width	float	Global	設定ファイルの HR_width 参照
jsonmanager	JSONManager	Global	外部ファイル読み込み
pos_up	Vector3	Local	上部デッドゾーン範囲オブジェクトの位置
pos_lw	Vector3	Local	下部デッドゾーン範囲オブジェクトの位置

8. HeartRate.cs

   このスクリプトは，初期化のために最初にそれぞれ一度だけ呼び出される Awake 関数と Start 関数，毎フレーム呼び出される Update 関数からなる．

Awake 関数では，毎秒フレーム数を 60fps に設定している．

Start 関数では，心拍数を受信するために，Node.js サーバーと websocket を使用して接続する．

Update 関数では，フレームが 15 フレーム毎に（60fps なので 0.25 秒毎に）Node.js サーバーから心拍数を受け取る．受け取った心拍数が null でないとき，心拍数を float 型に型変換し，スクリプトがアタッチされているオブジェクト（この場合 Circle）の transform を取得する．取得した transform 内の position の y 座標を float 型に変換した心拍数にして，オブジェクトに position を与える．また，心拍数の csv 出力と画面上に表示する．なお，この時出力される csv はプロジェクトフォルダ直下の TestSaveData.csv であるが，これはデバッグ用のデータなので必要に応じて中身を消去することを推奨する．

表 4.15　HeartRate.cs の変数一覧

変数名	型	スコープ	説明
sw	StreamWriter	Global	心拍数を csv 出力するための StreamWriter オブジェクト
ws	WebSocket	Global	心拍数を受信するための WebSocket オブジェクト
stackText	string	Global	受信した心拍数を受け取るための string 型変数
targetText	Text	Global	未使用
HRText	TextMeshProUGUI	Global	心拍数表示のための TextMeshPro オブジェクト
player	GameObject	Global	自機（飛行機）の GameObject
heartrate	HeartRate	Global	HR オブジェクト参照
HR	float	Local	受信した心拍数を float 型として格納するローカル変数
url	var	Local	WebSocket を接続するホストサーバー URL

9. LineTracker.cs

このスクリプトでは，ゲームのプレイ中に表示される，飛行機の軌跡を模した心拍数の履歴の表示を管理している．軌跡は LineRenderer を使用して描画される．LineRenderer に対して，3 次元座標（Vector3）の配列を渡すことで，各座標を結ぶ線が描画される．本スクリプトではこの配列の要素数は 100 としている．

Staart 関数では，軌跡描画のための LineRenderer オブジェクトを，「Line」というオブジェクト名から取得する．このため，軌跡を描画する LineRenderer オブジェクトは「Line」という名前でなければならない．次に，Line の初期化を行う．Line に渡すための 3 次元座標配列の各 x 座標を設定ファイルの spd ずつずらす．次に y 座標はすべて Start 関数実行時の心拍数にする．z 座標は使用しないため，任意の値としてすべて 0 とする．

Update 関数では，Line に与える 3 次元座標配列を更新することで，まるで Line が移動しているように見せている．0.25 秒毎に，心拍数を受け取り，3 次元座標配列の最初の 3 次元座標の y 座標を，心拍数から目標心拍数を引いた値とする．

表 4.16　LineTracker.cs の変数一覧

変数名	型	スコープ	説明
lr	LineRenderer	Global	飛行機の軌跡を描画するための LineRenderer オブジェクト
tmp	GameObject	Global	Unity 内から LineRenderer オブジェクトを受け取る
pos	Vector3[ ]	Global	LineRenderer に渡す 3 次元座標配列
max	int	Global	pos の要素数
ws	WebSocket	Global	心拍数を受信するための WebSocket オブジェクト
stackText	string	Global	受信した心拍数を受け取るための string 型変数
targetText	Text	Global	未使用
HR_tgt	float	Global	目標心拍数 外部設定ファイル（config.json）参照
HR	float	Global	心拍数
spd	float	Global	軌跡の描画間隔 外部設定ファイル（config.json）参照
heartrate	HeartRate	Global	HR オブジェクト参照
pos_a	var	Local	LineRenderer に与える座標を更新するために，一時的に座標を保管しておく変数

10. PlayerManager.cs

　このスクリプトでは，プレイヤーの自機（飛行機）の挙動と，その状態に応じたオブジェクトの配置や関数の呼び出しを管理している．

　Start 関数では様々な変数やオブジェクトの初期化を行っている．

　Update 関数では，自機の状態（主に位置）によって変動する status によってオブジェクトの配置や関数の呼び出しを行っている．

表 4.17　PlayerManager.cs の変数一覧

変数名	型	スコープ	説明
status	byte	Global	自機（飛行機）の位置を基準としたゲームの状態
startFrame	int	Global	ゲームプレイ画面（02_main）に遷移した時の経過フレーム数
time_flying	int	Global	水平飛行の合計時間 [s]
depTime	byte	Global	離陸にかける時間 [s]
grdTime	byte	Global	離陸滑走の時間 [s]
rotTime	byte	Global	離陸の際の回転にかける時間 [s]
time_s	byte	Global	未使用
dx	float	Global	離陸モーションの際の x 方向移動量
dy	float	Global	離陸モーションの際の y 方向移動量
HR_tgt	float	Global	目標心拍数 外部設定ファイル（config.json）参照
HR_width	float	Global	心拍数許容範囲 外部設定ファイル（config.json）参照
heartrate	HeartRate	Global	HR オブジェクト参照
scoremanager	ScoreManager	Global	ScoreManager スクリプト参照
jsonmanager	JSONManager	Global	jsonmanager スクリプト参照
DZ_Up	GameObject	Global	上部デッドゾーン範囲
DZ_Lw	GameObject	Global	下部デッドゾーン範囲
Line	GameObject	Global	LineRenderer オブジェクト
OverSpotter	GameObject	Global	飛行機が描画範囲外に出たことを知らせるオブジェクト
OS_airplane	GameObject	Global	飛行機が描画範囲外に出たことを知らせるオブジェクト
Background_land	GameObject	Global	背景オブジェクト（地上）
TextHRtgt	TextMeshProUGUI	Global	心拍数表示用のテキスト
TextMsg	TextMeshProUGUI	Global	未使用
frame	int	Local	ゲームプレイ画面（02_main）に遷移してからの経過フレーム数
cloud_obj	GameObject	Local	雲オブジェクト
myTransform	Transform	Local	スクリプトをアタッチしたオブジェクトの Transform
pos	Vector3	Local	myTransform の position

11. ScoreManager.cs

このスクリプトでは，ゲームプレイ中のスコアを管理している．

表 4.18　ScoreManager.cs の変数一覧

変数名	型	スコープ	説明
jsonmanager	JSONManager	Global	jsonmanager スクリプト参照
heartrate	HeartRate	Global	HR オブジェクト参照
playermanager	PlayerManager	Global	playermanager スクリプト参照
Text_score	TextMeshProUGUI	Global	スコア表示のためのテキストオブジェクト
score	int	Global	スコア
status	float	Global	playermanager の status 参照
t_run	float	Global	設定ファイルの t_run 参照

（※文責: 網干界）

4.8.2　動作の流れ

本アプリケーションの動作の流れを以下に示す.

スタート画面（00_start）

1. 00_start シーンをロードし，他のシーンをアンロードする
2. Button_00_goMeasure ボタンがクリックされると，01_MeasureHR シーンをロードする

心拍数計測画面（01_MeasureHR）

1. Button_01main ボタンがクリックされると，MeasureHRCoroutine 関数を呼び出し，通常心拍数を測定し，Button_01_goPlay ボタンを表示する
2. Button_01_goPlay ボタンがクリックされると，02_main シーンをロードする
3. Button_skip_conf ボタンがクリックされると，skipCanvas を表示する
4. Button_01_skip ボタンがクリックされると，02_main シーンをロードする
5. Button_skip_cancel ボタンもしくは Button_x ボタンがクリックされると，skipCanvas を非表示にする

ゲーム画面（02_main）

1. JsonUtility 関数により config.json を読み込む
2. config.json の spd 値に応じて背景オブジェクトを移動する
3. departure 関数によって飛行機オブジェクトを画面中央に移動する
4. HR が HR_tgt-HR_width 以上になると，雲オブジェクトを画面上部描画範囲外から画面下描画範囲外まで移動する
5. HR_width から設定したデッドゾーンを表示し，fly 関数で飛行機の高度やスコアの加算などの処理を行う
6. time_flying が t_run より大きくなると，デッドゾーンを非表示にする

（※文責: 熊坂賢人）

第 5 章　最後に

5.1　まとめ

本プロジェクトは「X-Reality とスマートウォッチを用いて身の回りの現象に接続し，自分の気持ちと世界をつなぐための情報リンクを作成する」を目標とするプロジェクトである．目標達成のために，X-Reality とスマートウォッチそれぞれに関する技術を習得する必要がある．その上で X-Reality とスマートウォッチを組み合わせ，自分の気持ちと世界をつなぐツールを作成することが課題である．また，X-Reality とスマートウォッチをどのように用いて，どのように自分の気持ちと世界を繋ぐのかを検討する必要がある．これを達成するために前期では SWG と ARG の二つに分かれてそれぞれ活動していたが，後期ではグループを統一して活動を行った．

（※文責: 小田涼平）

5.1.1　前期 SWG のまとめ

SWG では，気持ちの変化は身体データに影響を与えうると考え，「自分の気持ちを認識する」ために自分の本当の気持ちを身体データとしてリアルタイムで可視化することを目標とした．身体データとは，ウェアラブルデバイスを装着した際に得られる時系列なデータであり，心拍数，血中酸素濃度，消費カロリーなどが挙げられる．気持ちの変化を数値としてデータ化することで，自分にはわからない本当の気持ちをデータから読み取ることが出来ると考え，この目標を設計した．まずリアルタイムでスマートウォッチや活動量計のデータの可視化を試みた．ANT ＋対応のスマートウォッチや活動量計と専用のドングルを用いて，PC へのデータ送信を行った．また，データ解析には python を使用し，表示には Node.js を使用することで心拍数の可視化を可能とした．実際にマッサージや筋トレを行うことで心拍数を取得した．心拍数を取得するアプリケーションを作成し，実際にデータをとることに成功したことから目標は達成できたと思う．

（※文責: 小田涼平）

5.1.2　前期 ARG のまとめ

ARG では「気持ちを仮想現実へ飛ばし世界と繋ぐ」ためのツールの制作を行うことを目標とした．これは，SWG が入手した自分の気持ちを外側へアウトプットする事で今までになかった自分の表現をする方法を見つけることにつながる．そのためのツールとして X-Reality を用いることにした．AR は現実世界の映像の上に仮想世界を重ね合わせて体験できる技術であり，当プロジェクトの目標は「自分の気持ちと世界をつなぐ」である．自らの気持ちは他者から正確に観測することは不可能なものであるため，仮想のものであると言える．よって共通点を見出したことから X-Reality の中でも AR を用いることにした．しかし，AR に関する技術や知識が不足しており，中間発表会までに最終成果物のプロトタイプを開発するのは厳しいと考えた．そのため，前期はすべて AR アプリケーション開発技術の習得に充てることにした．AR アプリケーション開発の技術

習得のためにはアプリケーションの作成の一連の動作をしたほうがいいと考えたため，前期活動目標を「AR ダーツゲームの作成」にした．そのため，前期活動では Unity を使った開発技術の習得から始め，AR アプリケーション開発の基礎となる技術を身に着け，それを用いて AR ダーツゲームの開発をすることができた．前期の目標である「AR ダーツゲームの作成」を達成することができ，AR アプリケーション開発技術の習得をすることができたことから前期の目標は達成できたと言える．

（※文責: 小田涼平）

5.1.3 後期のまとめ

先ほども述べたように前期では SWG，ARG に分かれて活動を行った．そこで習得した技術，具体的には前期 ARG で習得した Unity を用いたゲーム開発の技術と，前期 SWG で使用したスマートウォッチを用いた心拍数を可視化する技術を統合し，後期では１つのアプリケーションを作成することにした．

後期の最初にスマートウォッチの技術を用いたアプリケーションを作成することに決めた．スマートウォッチの運動中の心拍数について調べていると心拍数には運動し始めの時に大幅に上昇し，一定まで上昇するとある一定の心拍数付近を上下し，運動が終わると徐々に心拍数が下降することが分かった．これは飛行機の動きの最初に大きく高度を上昇させる「離陸」，一定まで上昇するとその付近の高度で停滞する「飛行」，そして最後に徐々に高度を下降させる「着陸」の部分について酷似していることが分かった．そのため心拍数を飛行機の高度に見立てたアプリケーションを作成することに決定した．また，ダンベルを持ち上げたり，スクワットを行ったりなどの動きが一定でないものよりも，ウォーキングやランニングなどのような一定の動きを繰り返すもののほうが心拍数が安定しやすいため適していると考えた．また，ランニングマシンを使用中は特にやることがなく暇になりがちである．そのためランニングマシンの暇を解消しつつ適切な運動ができるようなアプリケーションを作成しようと決めた．

後期ではグループが合体して最終成果物が確定していなかったので案出しをし，その後作業分担を前期の SWG と ARG にわかれて行った．SWG は実際に運動を行いデータの採取，アプリケーションデザインなどを行い，ARG は Unity を用いたデータ採取用アプリケーション開発，最終成果物のアプリケーション開発を中心に行った．

成果物は心拍数を飛行機の高度と見立て，楽しみながら適切な運動ができるようにサポートしたアプリ「HEARTBEAT AIRLINE」を作成した．これは主に「離陸」,「上昇」,「飛行」の 3 つのシーンの流れからなるアプリケーションである．「離陸」ではプレイヤーが実際に運動を始めるタイミングを示している．「上昇」では自分で設定した適切な運動を行えると思った心拍数付近へと近づくまで行われている．「飛行」ではプレイヤーが設定した時間「飛行」が行われ，プレイヤーが設定した心拍数± 10 の値の時に得点が追加される．また，ハイスコアも確認することができる．これによりランニングマシン使用中に自分がどのような運動をしているか可視化しやすくなり，適切な運動ができるようにサポートすることができる．

（※文責: 小田涼平）

5.2 結果

本プロジェクトは「5.1 まとめ」でも述べたように「X-Reality とスマートウォッチを用いて身の回りの現象に接続し，自分の気持ちと世界をつなぐための情報リンクを作成する」を目標とするプロジェクトである．その目標を達成するために本プロジェクトでは「1.2 目的」でも述べたように誰でも気軽に運動を始められ，運動を継続してもらえるようにするためのゲーム開発を行った．開発したゲームの名前は「4.4 成果物」「5.1 まとめ」でも述べたように「HEARTBEAT AIRLINE」とした．そのゲームは心拍数に応じた飛行機を既定の範囲に維持させることでスコアが加算されるという仕様となっている．

「HEARTBEAT AIRLINE」を利用することで運動習慣を維持すると思われる要因は以下の三つである．

1. 心拍数を用いることで自分の体力がどの程度かを調べることができる
2. 既定の範囲を自分で設定し難易度を変えることができる
3. 走ること以外にも注意を向けるため退屈はしない

「1. 心拍数を用いることで自分の体力がどの程度かを調べることができる」では，運動を久しぶりにする人自身の体力を知り指標とすることができ，運動を続けることで心拍数の上昇傾向を可視化して知ることにより以前の自分と比較することができる．そのためより自身の成長を実感することができる機能となっている．

「2. 既定の範囲を自分で設定し難易度を変えることができる」では，難易度のマンネリ化を防ぐ，「3. 走ること以外にも注意を向けるため退屈はしない」では，心拍数を投影した飛行機をスコアが加算される範囲内に留める必要があるためランニング中の退屈を凌ぐ，と「HEARTBEAT AIRLINE」の利用者を楽しませるゲーム性となっている．

これらの機能性とゲーム性により「1.3 従来例」でも述べた「ながら運動」をして運動習慣の維持ができるのではないかと考えられる．

そして，当初の本プロジェクトの目標については，自身の身体データを PC という身の回りのものに接続し，心拍数の鼓動の激しさという気持ちを我々の目に見えるものにつなげる情報リンクを「HEARTBEAT AIRLINE」というゲームに落とし込むことで作成できたため達成できたのではないかと考える．

（※文責: 宮崎隼）

5.3 結論

本年度の本プロジェクトにおける最終目標であった運動時の心拍数を用いたゲームを開発できた. しかし, 本プロジェクト内においては実験や発表会での実演及びデバッグ作業において 10 回程度利用したが, これらを用いてユーザーの興味・関心を引き出すことができたかを客観的に評価できなかった. また, プロジェクト時間外に各メンバーが自発的にこのゲームを使用した回数は 0 回であった.

（※文責: 網干界）

5.4　結論に至った理由

5.4.1　各メンバーの結論に至った理由

ここではメンバー１人１人が結論に至った理由を述べる。

（※文責: 丹野陽翔）

心拍数を用いたゲームを開発することで目標は達成できた．しかし，自分がユーザーでもあるという認識が十分でなく，プロジェクト外で使用することはなかった。このため開発したゲームは、目的であるユーザーの興味・関心を引き出し，運動をはじめとする活動が楽しくなるようサポートしたとはいえなかった。このため失敗という結論に至ると考えた．

（※文責: 八木田光）

本プロジェクトでは技術的な面では目標を達成できた．しかしながらユーザーの目線に立てておらず使いづらい点が残るアプリケーションとなり，結果としてユーザーの興味を引くものにはならなかった．

（※文責: 小田涼平）

技術先行のアプリケーション制作であり，ユーザーの興味・関心を引き出すアイデアの検討が不十分だった．

（※文責: 森皐太）

本年度の本プロジェクトでは，運動時の心拍数を用いたゲーム開発を行った．しかし，ユーザー目線から見た開発が不十分であったため，このような結論となった．

（※文責: 網干界）

本プロジェクトにおいて，アプリケーションの完成という大きな目標を達成することはできたが，自分たちはデモや実験で使用することで，ユーザーの意見を理解できると誤認してしまっていた．それによって時間外での使用をおこたってしまったことが一番の問題点だったと今になって思う．またゲームを開発していくうちにそのアプリケーションを主観的に捉えてしまったことでより，ユーザー目線でのアプリケーション開発にはなっていなかった．これらのことが今回の結論に至った．

（※文責: 安澤龍生）

本プロジェクトでは，技術を使うことを意識しすぎて，ユーザーが楽しく利用するという点をあまり考慮できていなかった．今後の展望として，アプリケーションの試用期間を長く取り，実際に使用したいと思えるようなアプリケーションを作成したいと考える．

（※文責: 熊坂賢人）

アプリケーションの完成が本プロジェクトの最終目標だと考えてしまい，制作したアプリケーションを実際に使用して感想・改善点をフィードバックするという工程を失念してしまっていた．これによって完成したのは開発者目線のみのアプリケーションであり，ユーザーの興味・関心を引

き出すことができるアプリケーションであるかは判断することができない．以上のことから上記の結論に至った．

（※文責: 川上龍仁）

私を含め, 本プロジェクトメンバー全員がゲームの開発者であると同時に, このアプリのユーザーでもあった. 本プロジェクトはアプリの使いずらさであったり, 知りたい情報を見ることが出来ないなどユーザーとしての視点が欠けていたために満足のいく結果を残すことが出来なかった. したがってユーザーフレンドリーなアプリケーションを作成していくことを今後の課題とすべきだろう．

（※文責: 丹野陽翔）

アプリケーションとして十分に完成させることができた．しかし，評価者のフィードバックを反映していなかったり，自分が不十分だと感じていても問題ないと過信していたところがあった．また，時間配分と作業効率の面でも問題があった．UI やユーザーへの達成感の提供など，細かな機能に改善できる余地があると考えた．

（※文責: 尾崎篤史）

ランニングマシンの暇つぶしをするには，このアプリケーションは音楽を聴くことよりも面白いと感じる価値が低いからである．我々はユーザーの視点を過失して開発を行ってきたことにより，ゲームの価値をあげることができなかった．しかし，ランニングマシンに固執せず，ジェンガなどの心拍数の変化を取れるゲームの一つの要素に組み込むことでこのままの状態でもより利用する価値が高まると考える．

（※文責: 宮崎隼）

5.4.2　成果発表会を通した今後の展望

成果発表会を通していくつもの質問や意見をいただいた．それらをもとに考えた今後の展望を以下に述べる．

Q：XR はどこに行ったのか．
A：実装ができなかった．そのため XR の要素を AR で付け加えるならスマートグラス，VR なら Meta Quest2 など頭部に装着する運動を行う際に邪魔にならない端末を使用することで実装する．

Q：他の運動にも使用できそう．
A：今回はランニングマシン用のアプリを開発したのでそれ以外の運動，例えば未来大学のトレーニングルームを活性化させることを目的としたトレーニングルームにあるすべての筋トレ器具用に対応したアプリを作成する．運動ではないが，本プロジェクト内でもトランプや人狼などの心理ゲームなどにも応用することが出来るのではないかという案も見受けられた．

Q：着陸の動作はないのか．

A：現時点では作成できなかったので運動後に心拍数を下げる動きと連動させるような形式で着陸のシーンを追加する．

Q：心拍数の変更がしづらそう．
A：心拍数の変更を json ファイルの変更ではなく，アプリ内で変更できるように改善する．

Q：手持ちのメモを見ている学生がいる．アプリケーションの作成に至った背景が不明だった．
A：発表の準備不足が招いた結果なので発表の原稿を暗記し見ないようにする．この問題は，中間発表の時から指摘されていた点であり，今後さらなる注意と入念な準備を怠らないようにする．アプリケーションの作成に至った背景が不明だった点については，アプリケーションの説明を簡潔にスライドにまとめる際に，考慮が足りていなかったため，説明の際に作成に至った背景をスライドに追加し，発表でもそれについて簡潔にまとめたうえで説明をする．

Q：ルームランナー側の UI（Unity 画面の表示）は，スクリーンを使わない方法は考えられるでしょうか？（例えば，ウェアラブルデバイスに表示するとか．）
A：現時点ではスクリーンを使用しない方法は不可能だが，Apple Watch のようなスマートウォッチや Nreal Air のようなスマートグラスなどのようなウェアラブルデバイスでアプリケーションを使用できるようにすることで，ユーザーがアプリケーションを利用しやすくなると考えられる．そのため今後の課題として別のデバイスでも利用できるようにすることを検討する．

Q：スコアに応じて報酬が貰えるとかの機能とか付けたら良いと思いました．
A：スコアやプレイした回数，継続日数などに応じて動かしてる飛行機の見た目や色，画面の背景を追加し，ユーザーが変更できるようにすることでアプリケーションの継続的な利用を促す機能を追加する．

Q：ゲームオーバーなどで，ユーザーに健康を促せるような表示があるとさらにグッと良くなると思いました．
A：最後にリザルト画面を表示させ，そこに今回のスコア，ハイスコア，ユーザーへの一言，アドバイスなどを掲載し，フィードバックをすることでユーザーがより正しい運動を行うことができるような機能を追加する．

Q：ランニングマシン中の暇を解消するには少し物足りなさを感じた．
A：運動中のシーンやデッドゾーンに入っているかなどによって適切な音楽をかけたり，デッドゾーンに入った際にただ得点が入らなくなるだけではなく，画面を赤く点滅させ焦らせることでより臨場感をもって楽しんでもらえるようなアプリケーションにする．

Q: デモンストレーションで機体が想定と違う部分があった．
A: 成果発表会のデモンストレーションでは実際にプレイしている様子を google meet の共有機能を使用して共有していたのでその分のラグが生じており，それが想定と違う動きになったと思われる．実際に操作するうえでは問題がないと考える．

Q: 個人的にその斬新なゲームは健康面を考えるとよいゲームだと思いますが私は積極的にやり

たいようなゲームではないと思いました.

A: これは私たちも少し感じていた．5.3 の結論，5.4 の結論に至った理由で述べられている通り，私たちプロジェクトメンバー全員が開発者であると同時に，このアプリのユーザーであるという認識が欠けていた．そのためアプリの使いづらさや，知りたい情報を見ることができないなどユーザーの視点が足りておらず，技術的には成功していてもこのような積極的にやりたいと思えないアプリケーションになってしまった．そのためアプリの使いづらさであったり，知りたい情報を見ることなどを改善しユーザーにとって使いやすく必要な情報を取得できるようにする．

（※文責: 尾崎篤史）

参考文献

[1] 水野映子, “コロナ禍での運動不足問題を振り返る,” 第一生命経済研究所, 2022 年 9 月 https://www.dlri.co.jp/files/ld/205362.pdf, （最終閲覧日:2023-01-17）

[2] 任天堂, ”リングフィット アドベンチャー,” https://www.nintendo.co.jp/ring/, （最終閲覧日:2023-01-17）

[3] DNA, ”心拍数によって演出が変化するインタラクティブなホラーゲーム「Nevermind」,” https://dailynewsagency.com/2014/03/04/this-video-game-knows-when-9ux/, （最終閲覧日:2022-05-25）

[4] 若杉美歩 巻野雄介, ”ハンドマッサージにおける実施者の手の温度が受け手に与える影響,” 日本赤十字豊田看護大学紀要, 2020, 15 巻 1 号, p.25-33.

[5] じん, ”心拍数を配信画面に載せる方法！！,” https://tokaisodachi.com/archives/2335, （最終閲覧日:2023-01-17）

[6] TIME SPACE by KDDI, ”注目の「XR」（クロスリアリティ）とは？ VR, AR, MR との違いと最新事例を紹介,” https://time-space.kddi.com/ict-keywords/20180816/2406, （最終閲覧日:2022-05-25）

[7] 理化学研究所, ”もう 1 つの現実を体験する「代替現実システム」を開発,” https://www.riken.jp/press/2012/20120621_2/, （最終閲覧日:2022-01-26）

[8] Unity Technologies, “AR plane manager — AR Foundation — 4.0.12,” https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.0/manual/plane-manager.html, （最終閲覧日:2022-05-25）

[9] Unity Technologies, “AR tracked image manager — AR Foundation — 4.0.12,” https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.0/manual/tracked-image-manager.html, （最終閲覧日:2022-06-03）

[10] Unity Technologies, “プレハブ - Unity マニュアル,” https://docs.unity3d.com/ja/2021.3/Manual/Prefabs.html, （最終閲覧日:2022-05-25）

[11] Unity Technologies,“コライダー - Unity マニュアル,” https://docs.unity3d.com/ja/2021.3/Manual/CollidersOverview.html, （最終閲覧日:2022-06-17）

[12] Unity Technologies,“Rigidbody-isKinematic - Unity スクリプトリファレンス,” https://docs.unity3d.com/ja/2021.3/ScriptReference/Rigidbody-isKinematic.html, （最終閲覧日:2022-06-17）

[13] modis, “GitHub とは？使い方や知っておきたい知識を解説！,” https://www.modis.co.jp/candidate/insight/column_30, （最終閲覧日:2023-01-11）

[14] RalaCode, “GitHub で Organization を作る,” https://ralacode.com/blog/post/github-organization/, （最終閲覧日:2023-01-15）

[15] たなかゆう, “GitHub と GitHub Desktop を使った小規模チーム開発 - tanaka’s Programming Memo,” https://am1tanaka.hatenablog.com/entry/2015/11/06/130120#WinMerge, （最終閲覧日:2022-06-10）

[16] たいらのエンジニアノート, “GitHub Desktop をインストールするやり方を解説しま

す," https://www.tairaengineer-note.com/github-desktop-install/, (最終閲覧日:2022-06-10)

[17] PENGIN, "【必見】 GitHub Desktop のインストールから使い方まで徹底管理！," https://pengi-n.co.jp/blog/github-desktop/, (最終閲覧日:2022-06-10)

[18] hkomo746, "Unity プロジェクトのバージョン管理には GitHub Desktop を使おう," Qiita, https://qiita.com/hkomo746/items/a4f99261868069f6eeb5, (最終閲覧日:2022-06-10)

[19] Garmin Canada Inc., "ANT+ BASICS," *ANT+ Basics - THIS IS ANT*, https://www.thisisant.com/developer/ant-plus/ant-plus-basics/ , (最終閲覧日:2023-01-17)

[20] Garmin Canada Inc., "ANT+ DEVICE PROFILES," *ANT+ Device Profiles - THIS IS ANT*, https://www.thisisant.com/developer/ant-plus/device-profiles , (最終閲覧日:2023-01-17)

[21] Garmin Canada Inc., "ANT / ANT+ DEFINED," *ANT / ANT+ Defined - THIS IS ANT*, https://www.thisisant.com/developer/ant-plus/ant-antplus-defined/ , (最終閲覧日:2023-01-17)

[22] Ian Fette, Alexey Melnikov, "The WebSocket Protocol," *RFC 6455: The WebSocket Protocol*, https://www.rfc-editor.org/rfc/rfc6455 , (最終閲覧日:2023-01-17)

[23] sta, "websocket-sharp," https://github.com/sta/websocket-sharp , (最終閲覧日:2023-01-17)

[24] NuGet, "WebSocketSharp.Standard," https://www.nuget.org/packages/WebSocketSharp.Standard/ , (最終閲覧日:2023-01-17)

[25] はなちる, "【Unity】websocket-sharp を用いてリアルタイムな通信をしてみる," https://www.hanachiru-blog.com/entry/2019/09/29/222228 , (最終閲覧日:2023-01-17)

[26] 大石 啓明, "Unity で WebSocket を使用する," https://qiita.com/oishihiroaki/items/bb2977c72052f5dd5bd9 , (最終閲覧日:2023-01-17)

[27] えど, "Unity で WebSocket を使ってブラウザと通信する," https://edom18.hateblo.jp/entry/2021/01/25/094602 , (最終閲覧日:2023-01-17)

[28] npaka, "WebSocket の使い方 - Unity," https://note.com/npaka/n/n3df3f9201f26 , (最終閲覧日:2023-01-17)

[29] Unity Technologies, "シーン," シーン - *Unity* マニュアル, https://docs.unity3d.com/ja/2021.3/Manual/CreatingScenes.html , (最終閲覧日:2023-01-17)

[30] Unity Technologies, "SceneManager," *Unity - Scripting API: SceneManager*, https://docs.unity3d.com/2021.3/Documentation/ScriptReference/SceneManagement.SceneManager.html , (最終閲覧日:2023-01-17)

[31] Unity Technologies, "SceneManager.LoadScene," *Unity - Scripting API: SceneManagement.SceneManager.LoadScene*, https://docs.unity3d.com/2021.3/Documentation/ScriptReference/SceneManagement.SceneManager.LoadScene.html , (最終閲覧日:2023-01-17)

[32] Unity Technologies, "SceneManager.UnloadScene," *Unity - Scripting API: SceneManagement.SceneManager.UnloadScene*, https://docs.unity3d.com/2021.3/Documentation/ScriptReference/SceneManagement.SceneManager.UnloadScene.html , (最終閲覧日:2023-01-17)

[33] Unity Technologies, “Button,” *Unity - Scripting API: Button*, https://docs.unity3d.com/2021.3/Documentation/ScriptReference/UIElements.Button.html , （最終閲覧日:2023-01-17）

[34] Unity Technologies, “Transform,” *Unity - Scripting API: Transform*, https://docs.unity3d.com/2021.3/Documentation/ScriptReference/Transform.html , （最終閲覧日:2023-01-17）

[35] Unity Technologies, “Quaternion.Euler,” *Unity - Scripting API: Quaternion.Euler*, https://docs.unity3d.com/2021.3/Documentation/ScriptReference/Quaternion.Euler.html , （最終閲覧日:2023-01-17）

[36] Unity Technologies, “LineRenderer,” *Unity - Scripting API: LineRenderer*, https://docs.unity3d.com/2021.3/Documentation/ScriptReference/LineRenderer.html , （最終閲覧日:2023-01-17）

[37] Unity Technologies, “JsonUtility,” *Unity - Scripting API: JsonUtility*, https://docs.unity3d.com/2021.3/Documentation/ScriptReference/JsonUtility.html , （最終閲覧日:2023-01-17）

[38] RIGHTCODE, “【Unity】ARFoundation 入門～機能解説から平面検知の実装まで～,” https://rightcode.co.jp/blog/information-technology/unity-ar-foundation-introduction, （最終閲覧日:2022-05-25）

付録 A　デバイスを選定した際の表

SWG でスマートウォッチや活動量計の選定をした際に作成した表について，以下にまとめる．以下は前期初めに選定したスマートウォッチのリストである．

表 A.1　スマートウォッチリスト 1

機種名	心電図	心拍数	血圧	皮膚温 or 体温	血中酸素濃度
AppleWatch Serise6	○	○	×	×	○
AppleWatch Serise7	○	○	×	×	○
AppleWatch SE	×	○	×	×	×
Fitbit Charge 4	×	○	×	△ (晩のみ)	○
Fitbit Charge5	×	○	×	△ (晩のみ)	○
Fitbit Luxe	×	○	×	△ (晩のみ)	○
Fitbit Sense	×	○	×	△ (晩のみ)	○
Fitbit Versa3	×	○	×	×	○ (睡眠時)
Amazfit GTR 3 Pro	×	○	×	○ (精度は怪しい・腕の表面温度)	○
Amazfit GTS 2 mini	×	○	×	×	○
HUAWEI band 6	×	○	×	×	○
HUAWEI band 6 pro	×	○	×	○ (皮膚温・常時は不可)	○
HUAWEI Watch FIT	×	○	×	×	○
HUAWEI WATCH GT 3	×	○	×	○ (皮膚温・常時計測)	○
RIVERSONG Motive 3S	×	○	×	×	○
sony wena 3 metal	×	○	×	×	○
vivosmart 4	×	○	×	×	○
vivosmart 5 Black S/M	×	○	×	×	○
OPPO Band Style	×	○	×	×	○
SMART R NY-17	×	○	×	○	○
SMART WRISTBAND-W8	×	○	×	○	○
SUUNTO 5 PEAK	×	○	×	×	×
Xiaomi Mi smart band 5	×	○	×	×	×

表 A.2　スマートウォッチリスト 2

機種名	血糖値	呼吸数	windows へのデータ転送
AppleWatch Serise6	×	睡眠時	ヘルスケアからエクスポート可能 即時データ送信可能
AppleWatch Serise7	×	睡眠時	同上
AppleWatch SE	×	睡眠時	同上
Fitbit Charge 4	×	睡眠時	Fit bit アカウントから データエクスポート可能 エクスポートは 1 日 1 回のみ
Fitbit Charge5	×	睡眠時	同上
Fitbit Luxe	×	睡眠時	同上
Fitbit Sense	×	睡眠時	同上
Fitbit Versa3	×	睡眠時	同上
Amazfit GTR 3 Pro	×	呼吸効率 はわかる	複数デバイスでの同期可能
Amazfit GTS 2 mini	×	×	同上
HUAWEI band 6	×		血中酸素 24 時間モニタリング可能 ストレスレベルもモニタリング可能
HUAWEI band 6 pro	×	○	hitrava を使うことで可能だが, データのリクエストで数日かかる データが消えることがある
HUAWEI Watch FIT	×		同上
HUAWEI WATCH GT 3	×		同上
RIVERSONG Motive 3S	×	×	VeryFit アプリをヘルスケア連携、 後に zip ファイルに書き出しで可能
sony wena 3 metal	×	×	wena アプリをヘルスケア連携可能
vivosmart 4	×	×	Garmin Connect と同期 （ヘルスケアと共有可能）
vivosmart 5 Black S/M	×	×	同上
OPPO Band Style	×		不明
SMART R NY-17	×	×	不可
SMART WRISTBAND-W8	×	×	不可
SUUNTO 5 PEAK	×	×	suunto アプリから fit ファイル形式で転送可能
Xiaomi Mi smart band 5	×	×	アプリのバグが多い

表 A.3　スマートウォッチリスト 3

機種名	特徴	価格
AppleWatch Serise6	第三世代の光学式心拍センサ	\45.800
AppleWatch Serise7	iPhone と同期が可能	\48,800
AppleWatch SE	耐水性なし	\32,800
Fitbit Charge 4	バッテリーの持続およそ 7 日、 GPS による距離の計測も可能	\13,627
Fitbit Charge5	心拍数は，エクササイズ中は 1 秒間隔 通常時は 5 秒間隔で保存 常時皮膚温度を計測可能	\24,990
Fitbit Luxe		\16,355
Fitbit Sense	バッテリーは六日以上持続可能 (使用するモードによる) １分単位で詳細なデータを 7 日間保存 心拍数は，エクササイズ中は 1 秒間隔, 通常時は 5 秒間隔で保存 ストレスレベルを測定	\34,990
Fitbit Versa3	目標の心拍数で振動可能 心拍数は精度はある程度正確	\24,990
Amazfit GTR 3 Pro	ストレスモニタリング機能	\39,800
Amazfit GTS 2 mini	24 時間心拍数モニタリング 睡眠トラッキングが可能	\13,800
HUAWEI band 6	血中酸素 24 時間モニタリング可能	\8,580
HUAWEI band 6 pro	価格は中国サイト価格	\7,600
HUAWEI Watch FIT	血中酸素濃度が正確 iPhone とのペアリング可能	\15,180
HUAWEI WATCH GT 3		\31,680
RIVERSONG Motive 3S	50〜150 程度の bpm はある程度正確	\4,999
sony wena 3 metal	ソニー独自のアルゴリズムを搭載している	\36,300
vivosmart 4	心拍数は 24 時間可能，精度良い	\18,000
vivosmart 5 Black S/M	BodyBattery を計測可能 7 日間駆動するバッテリー	\19,800
OPPO Band Style		\4,480
SMART R NY-17	測定値の精度が低い	\12,100
SMART WRISTBAND-W8	楽天ランキング売り上げ１位 精度があまりよくない	\3,996
SUUNTO 5 PEAK	運動中の燃焼脂肪と炭水化物をグラムで表示	\43,890
Xiaomi Mi smart band 5	MiFitness アプリで毎日の健康データを確認 安いので大量に入荷することは可能	\4,490

以下のリストは AppleWacth では計測できない血圧と体温を計測するために購入した血圧計と体温計のリストである.

表 A.4　血圧計リスト

機種名	データの出力	価格
オムロン 上腕式血圧計 HEM-7281T	ヘルスケア経由での出力可能	\12,980
オムロン 上腕式血圧計 HEM-7600T-W	同上	\21,868
タニタ 上腕式血圧計 BP-224L-WH	同上	\10,233
オムロン 手首式血圧計 HEM-6233T	同上	\16,368
A & D デジタル血圧計 UB-533MR	同上	\3,722

表 A.5　体温計リスト

機種名	特徴	価格
けんおんくん MC-6800B	平均 15 秒で測定可能	\2,800
非接触体温計 quick(クイック)	複数人のデータをまとめて管理できる	\9,900
UT-201BLE	検温した結果を Bluetooth でアプリに送信	\8,047
UT-201BLE Plus	通信にかかる消費電力が上より 3 倍多い	\4,870
MC-652LC	平均 10 秒で測定可能	\3,673
ベビースマイル Pit+S-708	アプリがヘルスケアと同期できるか不明	\4,480

以下のリストは ANT+ 規格対応のスマートウォッチや心拍計のリストである.

表 A.6　ANT+ 対応スマートウォッチ・心拍計リスト

機種名	特徴	価格
vivosmart 4	スマートウォッチに比べて安い	\15,282
vivosmart 5	同上	\18,000
COOSPO HW807	光センサー心拍数想定技術を用いている	\4,800
iGP SPORT HR60	iGPSPORT クラウドプラットフォームで同期	\5,980
Garmin vivoactive 4	精度も価格も高い	\39,800
Garmin vivomove 3	同上	\29,645
XOSS 心拍計アームバンド ハートレートモニター	正確なモニター	\4,350
Wahoo TICKR FIT ハートレート	内蔵メモリー搭載	\9,783
POLAR OH1	胸につけるタイプに近い精度	\14,980
MIO SLICE	3 軸加速度センサー内蔵	\4,600

以下のリストは ANT+ 規格対応のドングルで PC に接続することで PC とスマートウォッチの同期が可能.

表 A.7　ANT+ 対応ドングルリスト

機種名	価格
【Allez】ANT+ USB ドングル スティック Zwift	\2,800
【CycleAnt】 ANT + USB ドングル Zwift	\3,500
CooSpo ANT+ USB ドングル USB 送信機受信機	\2,500
ThinkRider USB ANT + STICK ドングルアダプターワイヤレスレシーバー	\1,999

（※文責: 熊坂賢人）

付録 B　AR アプリケーションのスクリプト

ここでは,「3 章 AR アプリケーションの制作」で紹介した AR アプリケーションの開発の際に作成したスクリプトを掲載する.

（※文責: 森皐太）

B.1　平面検知アプリケーション

B.1.1　スクリプト

スクリプトの作成にあたり，次のサイトを参考にした [38].

Listing B.1　SpawnCube.cs

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.XR.ARFoundation;
5  using UnityEngine.XR.ARSubsystems;
6
7  public class SpawnCube : MonoBehaviour
8  {
9      private ARRaycastManager arRaycastManager;
10     [SerializeField] GameObject cube;
11     private List<ARRaycastHit> hits = new List<ARRaycastHit>();
12
13     void start()
14     {
15         arRaycastManager = GetComponent<ARRaycastManager>();
16     }
17
18     void Update()
19     {
20         if (Input.touchCount > 0)
21         {
22             Touch touch = Input.GetTouch(0);
23             if (touch.phase == TouchPhase.Began)
24             {
25                 if (arRaycastManager.Raycast(touch., position ,
                       hits TrackableType.PlaneWithinPolygon))
26                 {
27                     Pose hitPose = hits[0].pose;
28                     Instantiate(, cube hitPose., position hitPose.rotation);
29                 }
30             }
```

```
31         }
32     }
33 }
```

（※文責: 森皐太）

B.2 AR ダーツゲーム

B.2.1 スクリプト

Listing B.2 Dart.cs

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class Dart : MonoBehaviour
6  {
7      void OnCollisionEnter(Collision col)
8      {
9          if (col.gameObject.name == "Plate")
10         {
11             this.GetComponent<Rigidbody>().isKinematic = true;
12         }
13     }
14 }
```

（※文責: 森皐太）

Listing B.3 CapsuleScript02.cs

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.XR.ARFoundation;
5  using UnityEngine.XR.ARSubsystems;
6
7  public class CapsuleScript02 : MonoBehaviour
8  {
9      [SerializeField]
10     private GameObject Arrow_c;
11     float force = 0.5f;
12
13     void Update()
14     {
15         if (Input.touchCount == 0 || Input.GetTouch(0).phase != TouchPhase.
               Ended || Arrow_c == null)
16         {
```

```
17              return;
18          }
19
20          if (Input.touchCount > 0)
21          {
22              GameObject dart = Instantiate(Arrow_c, transform.position,
                    transform.rotation) as GameObject;
23
24              dart.GetComponent<Rigidbody>().AddForce(dart.transform.forward *
                    force, ForceMode.Impulse);
25          }
26      }
27  }
```

（※文責: 森皐太）

付録 C　中間発表会

ここでは中間発表会で用いたポスターと寄せられたフィードバックについて記載する．

（※文責: 八木田光）

C.1　ポスター

以下，中間発表会で使用したポスターである．

図 C.1　中間発表ポスター 1

図 C.2　中間発表ポスター 2

（※文責: 八木田光）

付録 D　実験用アプリ導入・利用マニュアル

D.1　導入 1（心拍数取得システム）

1. ANT+ 用ドングルのドライバ，Node をインストール下記 URL を参照 Node のバージョンは 16（16.18 以下）のみ動作確認済み.
   `https://docs.google.com/document/d/1l_VO6s10U7HgqnqvGCSu2se9k2Ej8WP4/edit?https://ja.overleaf.com/project/63a2c398a320e3fca71070ecusp=share_link&ouid=104266880405037002963&rtpof=true&sd=true`
2. プロジェクト共有フォルダ（project2022） ->1020107 ->1101 より「heartbeat-js.zip」を任意の場所にダウンロード.

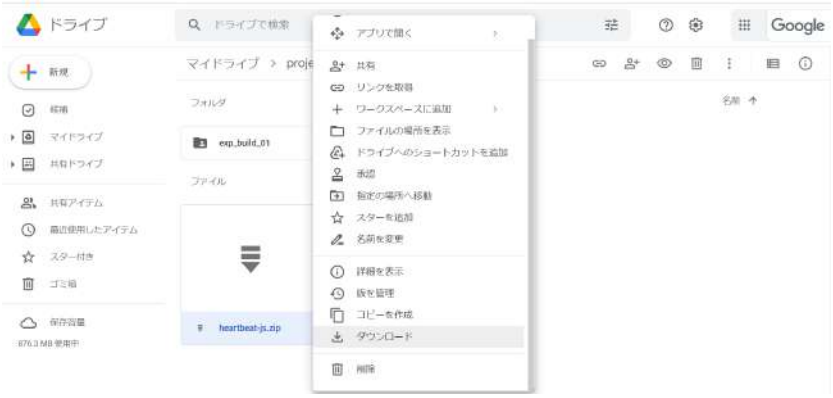

図 D.1

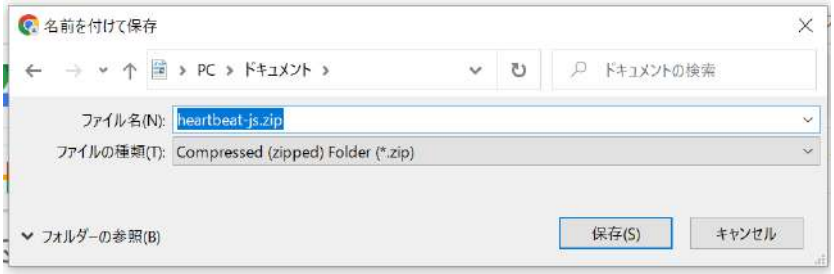

図 D.2

3. ダウンロードした zip ファイルを展開.

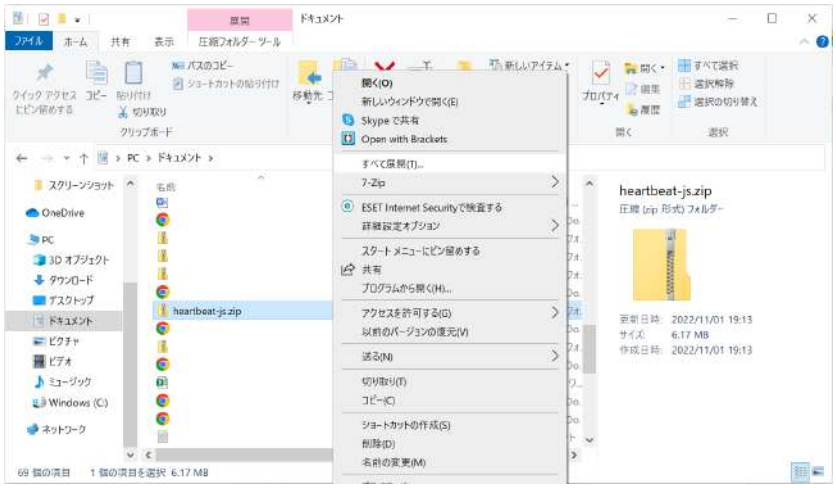

図 D.3

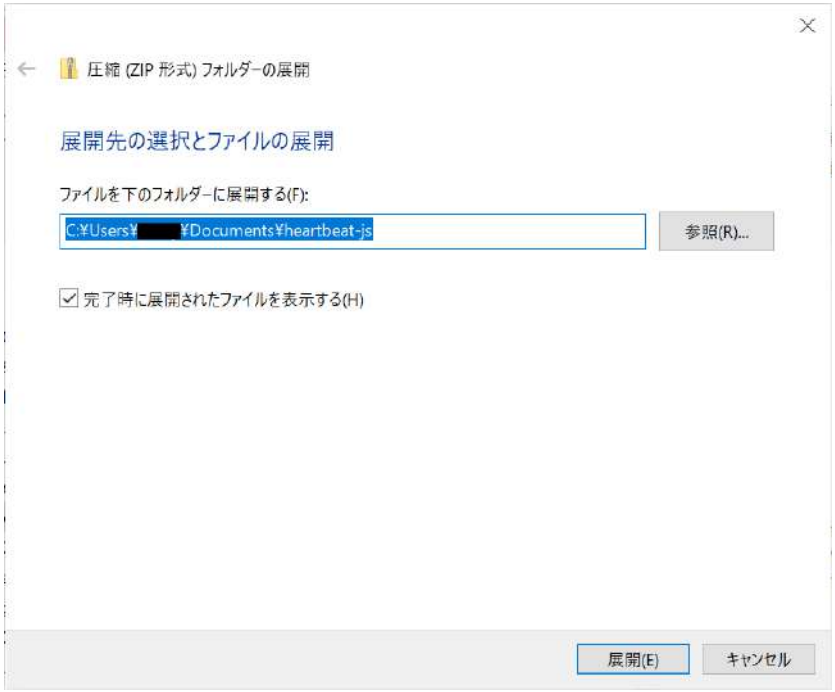

図 D.4

4. 展開したフォルダ内の「package.json」や「main.js」のあるフォルダまでのパスをコピー.

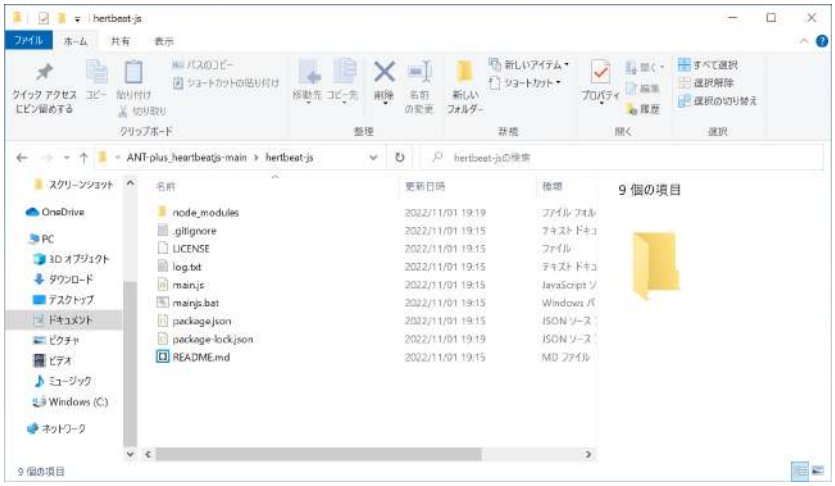

図 D.5

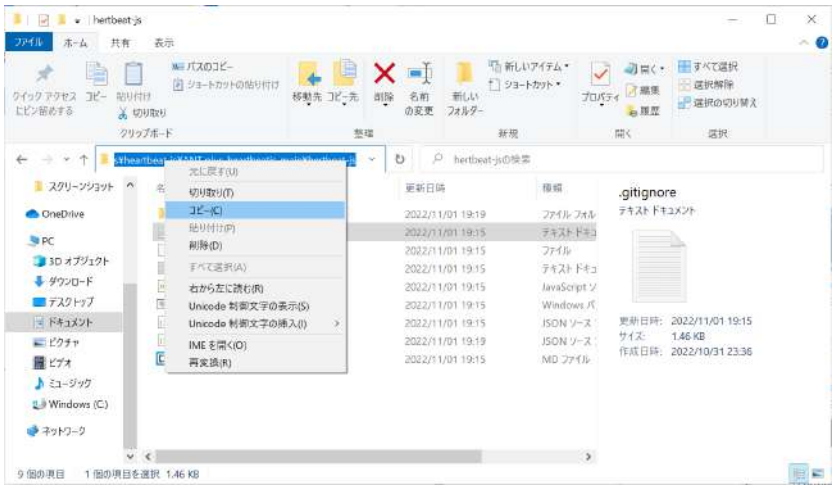

図 D.6

5. コマンドプロンプトを起動.

図 D.7

図 D.8

6. コマンドプロンプト上で，3. でコピーしたパスに移動
   >cd [コピーしたパス]

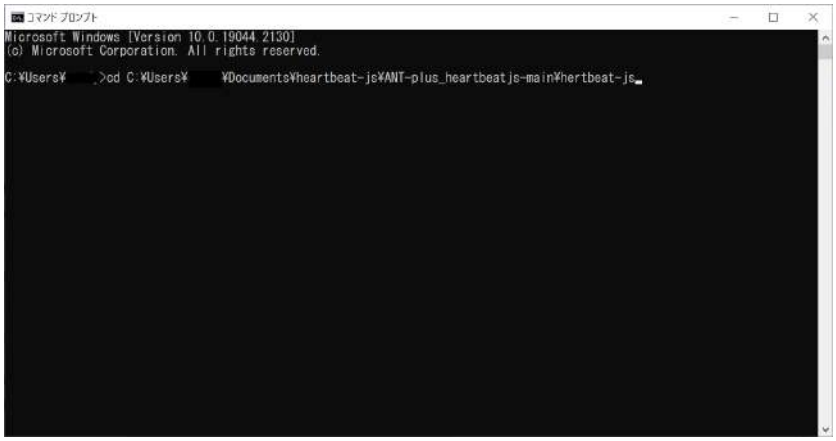

図 D.9

7. 下記コマンドでインストール.
   >npm install ant-plus

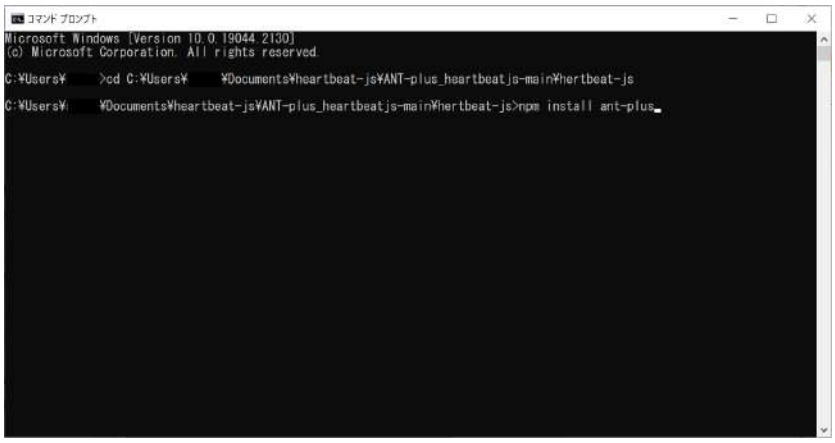

図 D.10

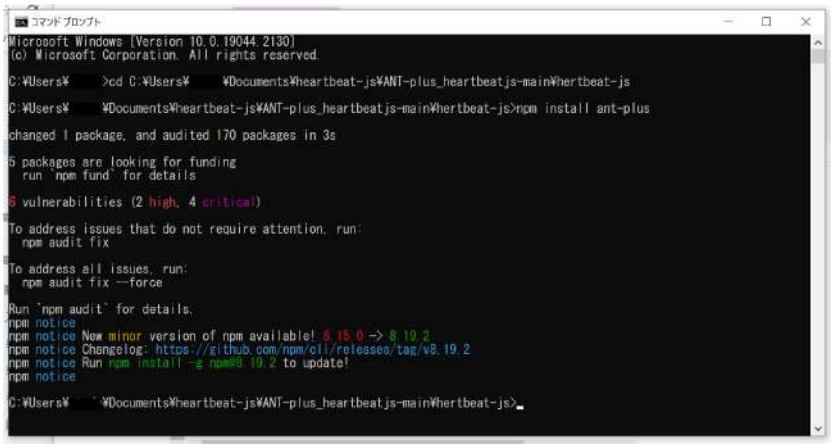

図 D.11

8. インストール完了後，コマンドプロンプトを閉じる.
9. スマートウォッチが心拍数送信モードになっていない場合は心拍数送信モードに設定する.
10. フォルダ内にある「mainjs.bat」を起動.

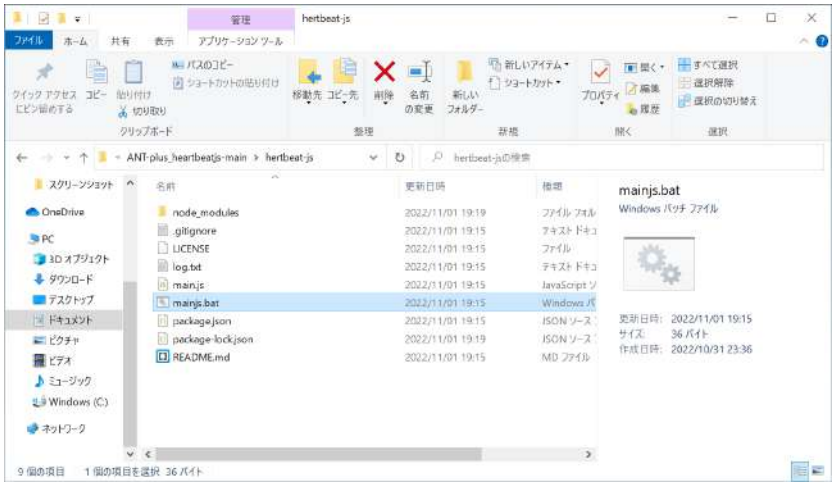

図 D.12

11. コマンドプロンプトが起動し，deviceID，心拍数，時間の情報が表示されていれば完了．

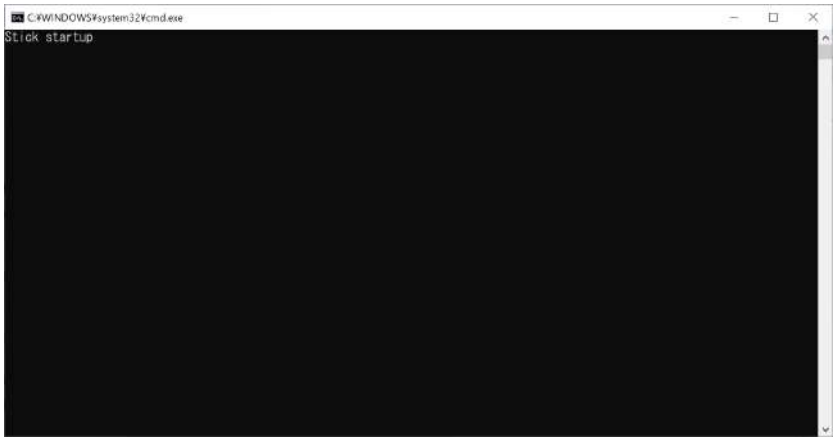

図 D.13

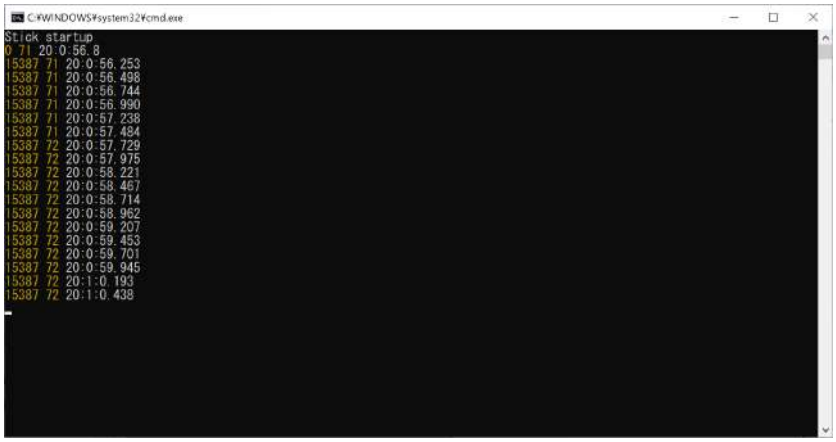

図 D.14

（※文責: 小田涼平）

D.2　導入 2（心拍数表示・保存アプリ）

1. プロジェクト共有フォルダ（project2022） ->1020107 ->1101 より「exp_build_01」フォルダを任意の場所にダウンロード.

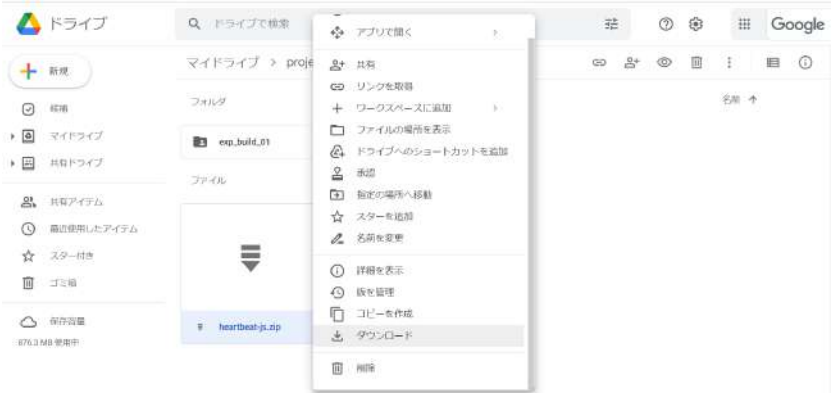

図 D.15

2. ダウンロードした zip ファイルを展開.

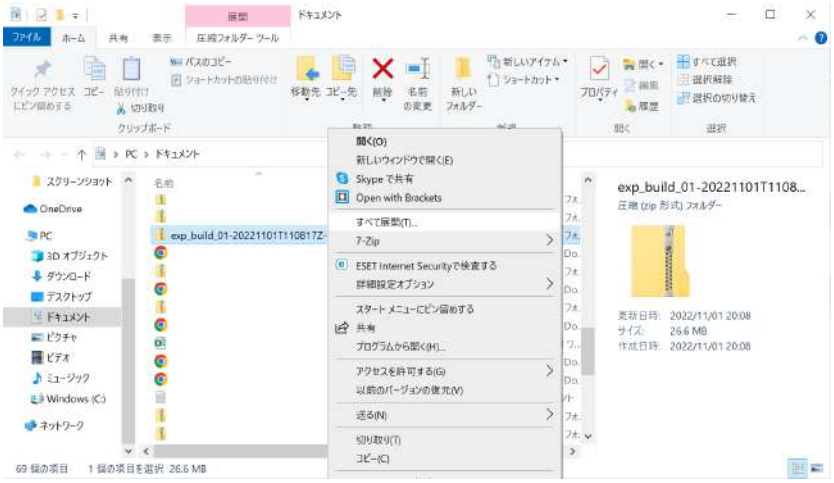

図 D.16

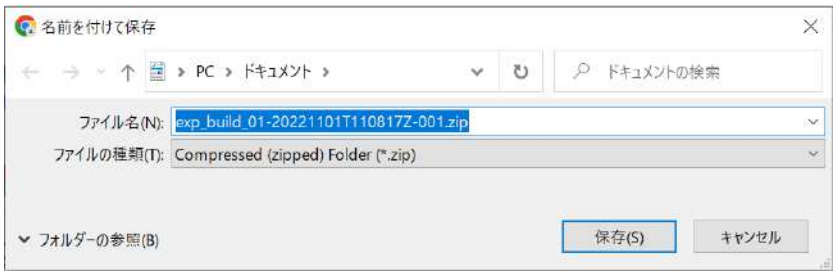

図 D.17

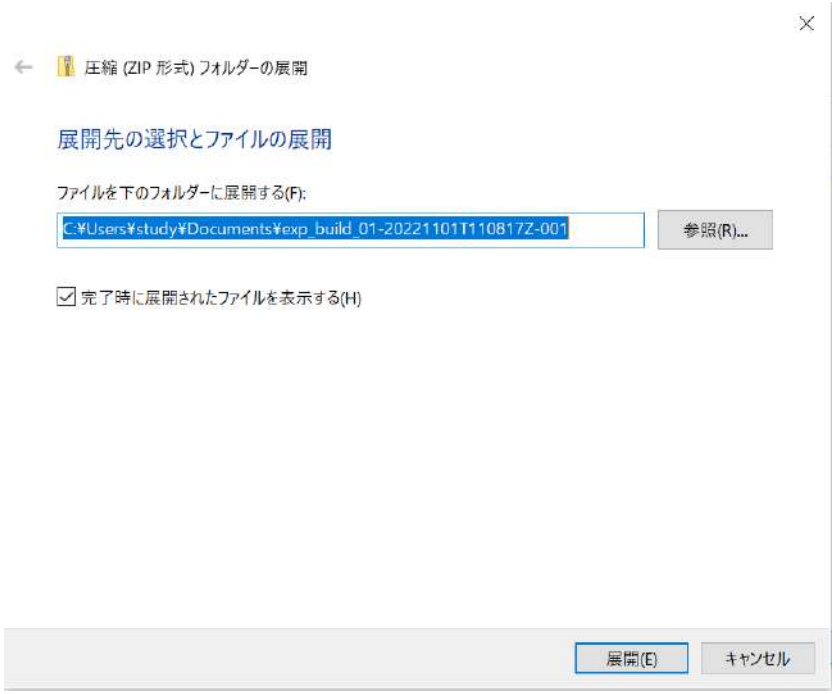

図 D.18

3. 導入 1-8 の「mainjs.bat」が起動されていなければ起動し，情報が受信出来ていることを確認.

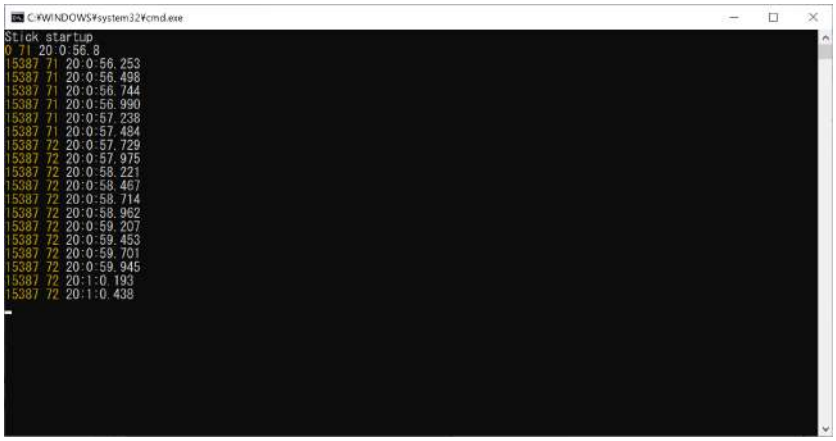

図 D.19

4. 2. で展開したフォルダ内の「ANT-plus.exe」を起動.

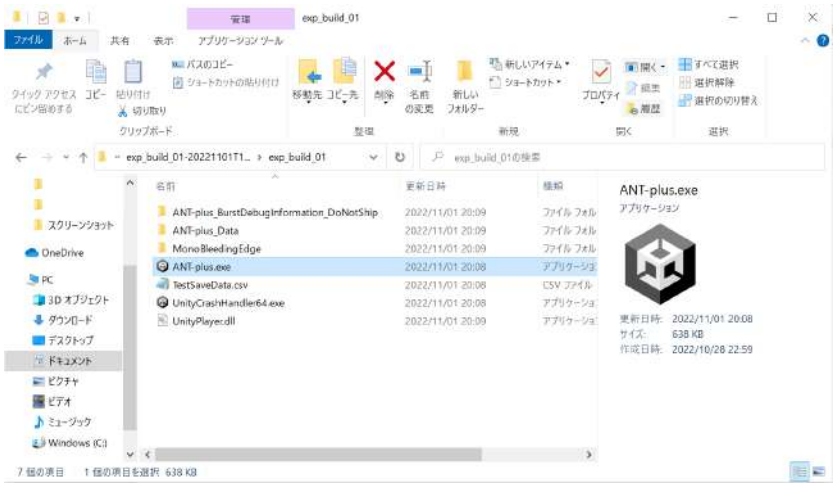

図 D.20

5. 現在の心拍数に対応した赤丸とグラフが表示されていれば正常に動作しているため完了.

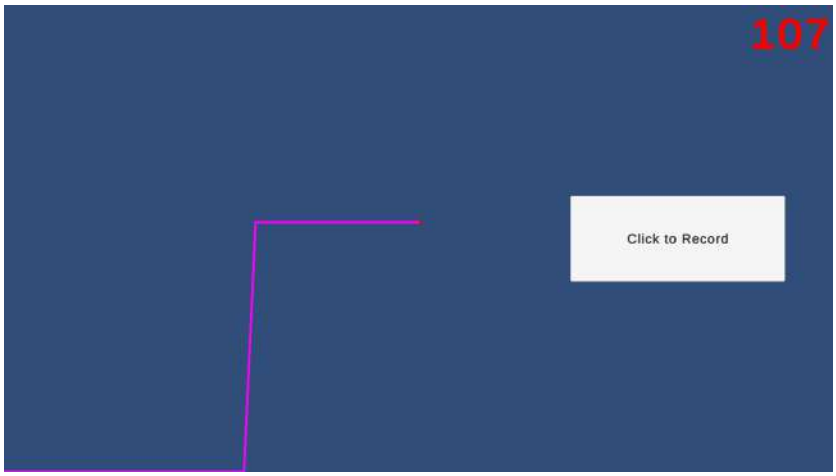

図 D.21

（※文責: 小田涼平）

D.3　利用方法

1. 導入 2 で起動したアプリが正常に動作していることを確認.

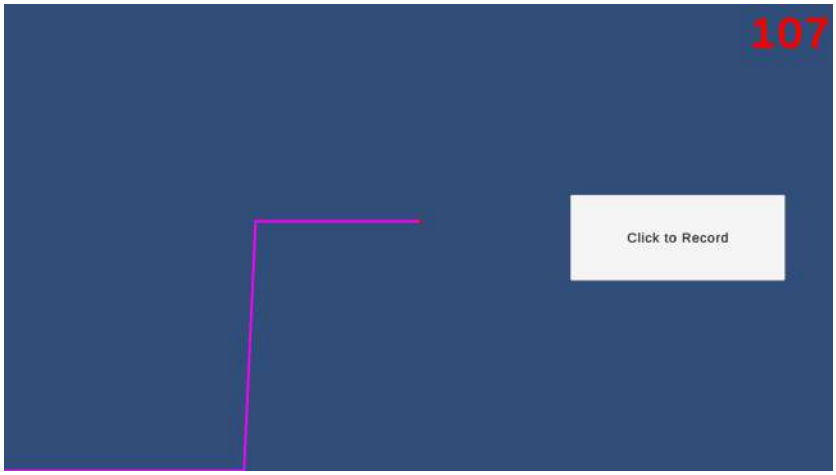

図 D.22

2. アプリ画面内の「Click to Record」ボタン（白色）を 1 回クリックすると，ボタンの色が赤色に，文言が「Recording HR ExpHR-yyMMdd-HHmmss.csv」に変化する．（yyMMdd-HHmmss はボタンを押した時の年月日-時分秒である）
   下図の状態で心拍数が記録されている．

図 D.23

   ボタンに表示されている「ExpHR-yyMMdd-HHmmss」と実験内容を別途記録しておくと後にデータと実験内容の紐づけがしやすい．
3. もう一度ボタンをクリックするとボタンの色と文言が最初の状態に戻る．

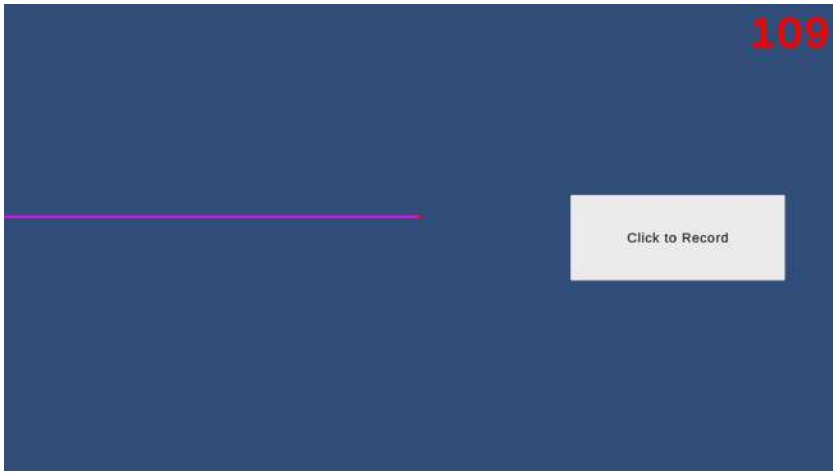

図 D.24

この時にデータの記録された csv が保存される．この状態では心拍数の記録はされない．

4. アプリのフォルダ（ANT-plus.exe のあるフォルダ）-> ANT-plus Data -> StreamingAssets -> csv フォルダ内に該当の csv ファイルが記録されている．

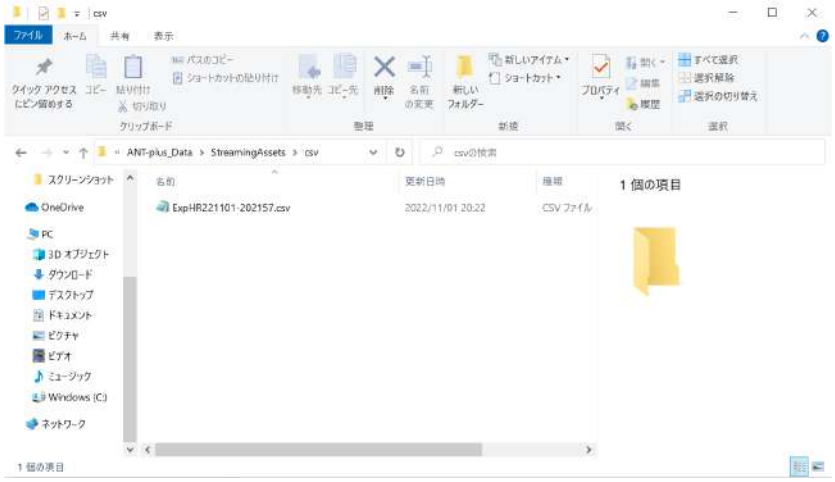

図 D.25

5. 終了するときは，アプリ内のボタンを白色の状態にしてから，ANT-plus.exe のタスクを終了（Win キーを押してタスクバーから終了等）．次に mainjs.bat を終了する．

補足 1 画面上の心拍数グラフは下限 0，上限 200 である．

補足 2 心拍数は毎秒 4 回更新・記録される．

補足 3 csv ファイルの保存はファイル名が同名の場合（同時刻に 2 回記録開始された場合），同じファイル内に追記される．

補足 4 csv ファイルの文字エンコーディングは UTF-8 である．

補足 5 csv ファイルの形式は，[時刻 (hh:mm:ss.ms)]， [心拍数] である．

（※文責: 八木田光）

D.4　困ったときは

- 心拍数が表示されない，アプリ内で Connecting... と表示される
  →スマートウォッチは心拍数送信モードに設定されていますか？
  解決しない場合は mainjs.bat と ANT-plus.exe を再起動してください.
  起動順序は「mainjs.bat」->「ANT-plus.exe」としてください.
  （スマートウォッチが心拍数送信モードに設定されていない場合，データを受信できません）
- アプリ内の心拍数が変化しない
  → mainjs.bat は正常に動作していますか？
  cmd 上に「Stick detached」と表示されている場合はスマートウォッチとドングルの距離を近づけてください.
  （ANT+ は 2.4GHz 帯を使用しています．一般的な Bluetooth や Wi-Fi 等の電波と干渉して接続が不安定になることがあります．距離を近づける，電波干渉の少ない場所に移動する等の解決策を試みてください）
  → mainjs.bat が正常に動作していない，強制終了している場合は mainjs.bat と ANT-plus.exe を共に再起動してください.（下図左が mainjs.bat，右が ANT-plus.exe）

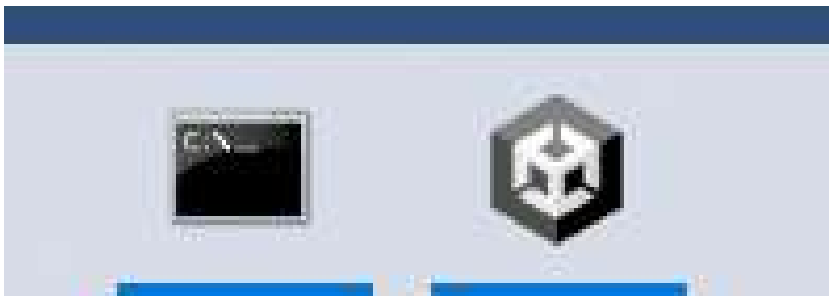

図 D.26

  起動順序は「mainjs.bat」->「ANT-plus.exe」としてください.
  （スマートウォッチとの再接続に失敗したり接続エラー等で心拍数取得システムが強制終了してしまう場合があります）
- 心拍数が 0 になる
  →スマートウォッチは正しく装着されていますか？
  スマートウォッチ上で心拍数が正しく表示されていますか？
  解決しない場合は mainjs.bat と ANT-plus.exe を共に再起動してください.
  起動順序は「mainjs.bat」->「ANT-plus.exe」としてください.
  （スマートウォッチでの心拍数が正常に取得できていないと 0 となる場合があります）
- 心拍数データ（.csv）が正常に記録されない
  →アプリ内のボタンの色は白色になっていますか？
  ボタンの色によって，白色（初期状態）->赤色（記録状態）->白色（記録保存・初期状態）といった遷移をします．ボタンが赤色の状態ではデータの保存が完了されず，もう一度クリックして白色に戻った時にデータの保存が行われます.
  →記録中にアクティブウィンドウがアプリ（ANT-plus.exe）になっていましたか？アプリがアクティブウィンドウになっていないと，アプリ，記録共に動作しません（バックグラウ

ンドでの動作はしません).
→（ANT-plus.exe のあるフォルダ）->ANT-plus_Data ->StreamingAssets フォルダ内に「csv」フォルダが存在していますか？
「csv」フォルダが存在しない場合は該当箇所に「csv」フォルダを作成してください.

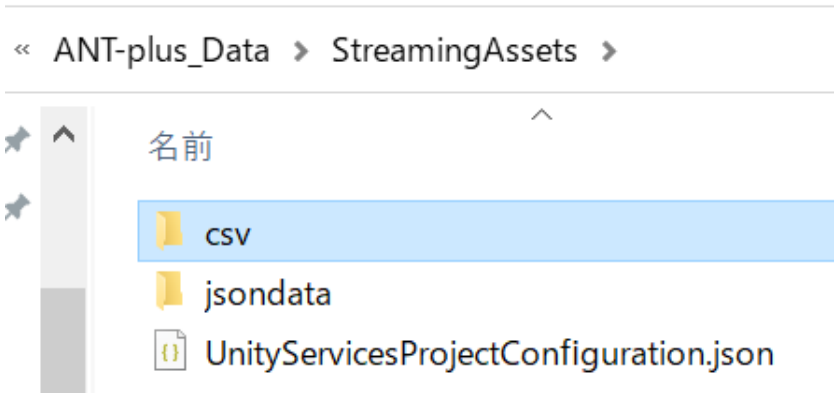

図 D.27

- 導入 1 において，インストールが上手くいかない・エラーになる→ cmd を「管理者として実行」から起動してみる．インストールする際にログインしているユーザーが管理者権限を保有していない場合，エラーとなることがあります．→ Node のバージョンを 16.17.1 にする→　node -v　上記コマンドで node のバージョンを確認する．16.17.1〜16.17.18 以外であった場合は Node のバージョンを 16.17.1 にする.
  1. コントロールパネル -> プログラムと機能　より Node.js のアンインストール
  2. 下記 URL より 16.17.1 の該当するインストーラをダウンロード.

`https://nodejs.org/download/release/v16.17.1/`

図 D.28

  3. 導入 1-1 に従いインストール
- mainjs.bat が起動しない→ ANT+ 用ドングルのドライバの再インストールを試みる→ PC の再起動を試みる→ Node がインストールされていることを確認する→　node -v　上記コマンドでバージョンが表示されれば Node がインストールされている→ Node のバージョンを確認する上記「導入 1 において，インストールが上手くいかない・エラーになる」の「Node のバージョンを 16.17.1 にする」を参照

（※文責: 八木田光）

付録 E　デザイン

ここでは，ゲームで用いたデザインを紹介する．
以下の画像はサイズを 3840 × 1080 にした背景である．

図 E.1　滑走路

図 E.2　空

次の画像は背景の滑走路にて付属する空港の画像である．

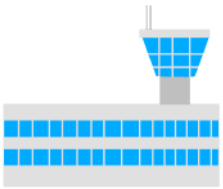

図 E.3　空港

次の画像はゲームでメインに利用した飛行機のデザインである．

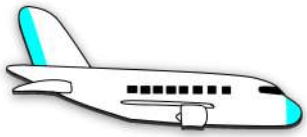

図 E.4　飛行機

次からの画像は制作時間上の関係でゲームに組み込めなかったデザインである.

図 E.5　気球

図 E.6　風船

図 E.7　麦わら帽子

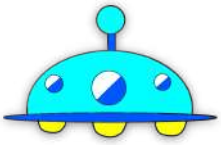

図 E.8　UFO

（※文責: 安澤龍生）

付録 F　成果発表会

ここでは成果発表会で用いたポスターと寄せられたフィードバックについて記載する．

F.1　ポスター

以下，成果発表会で使用したポスターである．

図 F.1　成果発表ポスター 1

図 F.2　成果発表ポスター 2

（※文責: 丹野陽翔）