令和 5 年度卒業研究報告書

OpenCV を活用したホッケーゲームの制作

情報技術科　佐藤　寛大

南館　玲来

指 導 教 員　菅野　研一

内容

第 1 章　はじめに

昨年の産技短展で、VR ゲームの人気が高まっていたが、小学生以下の児童に対しては VR 体験が斜視による悪影響を及ぼす可能性が指摘され、体験が制限されるという事態に至った。さらに、近年では子供たちの運動習慣の低下が深刻な問題として認識されている。このような状況を受け、私たちは子供たちが安全に楽しめると同時に、身体活動を促進することを目的とした新しいタイプのゲームの開発が必要だと考えた。

そのために、Unity と Python という二つの強力な開発ツールを使用して、室内で気軽に楽しめると同時に、健康的でインタラクティブなゲームを開発することに決めた。このゲームは、単に楽しいだけでなく、子供たちが日常的に運動をする習慣を身につけるきっかけを提供することを目指している。

この研究を通じて、私たちはテクノロジーが子供たちの健康と発達に及ぼす影響を積極的に形成し、改善することができると信じている。最終的には、このゲームが子供たちにとってだけでなく、家族全体が一緒に楽しめる健康的な娯楽の選択肢となることを望んでいる。

第 2 章　ゲーム構成

2.1 ゲーム企画

まず、ホッケーゲームを選んだ経緯は、子供たちが楽しめるものを考えたときに、ゲームセンターが思いついた。ゲームセンターの中にあるもので、デジタル技術を使用し、自分たちでも制作ができそうなホッケーゲームをテーマにゲームを企画した。

2.2 ゲームの仕様

・床にプロジェクターで投影した画面でホッケーゲームを行う.

・三脚に固定した Web カメラからフローリングワイパーの赤いポインタを読み取る.

・1 対 1 の対戦システムである.

・床での操作と PC 側の操作で分かれる.

・60 秒の時間制の対決にする.

・ゲームの流れを図　　に記載する

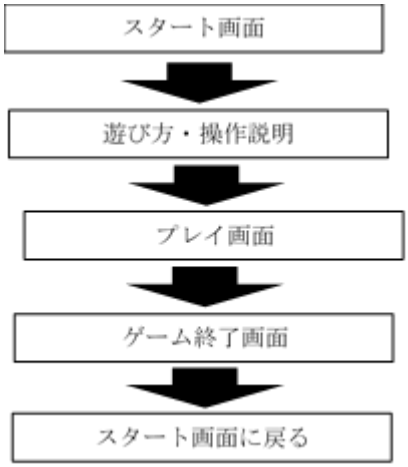


図 1　ゲームの流れ

2.3 キャラクター

2.3.1　キャラクターについて

キャラクターは生成 AI の「ChatGPT-4.0」を使用し,フローリングワイパーをモチーフにした「レッドくん」,キーボードをモチーフにした「キーボちゃん」という 2 体のキャラクターの画像を生成した.プレイヤーのキャラクターは PC 側がレッドくん.キーボード側がキーボちゃんである.ChatGPT-4.0 に「赤いフローリングを持ったキャラクターを描いてください。漫画風　かわいい」などとスクリプトを打ち込み送信し画像を生成した。

図 2　キーボちゃん

図 3　レッドくん

2.3.2　ChatGPT について

ChatGPT は、OpenAI 社が独自に開発した「GPT」と呼ばれる言語モデルを利用している。GPT は LLM（Large Language Models）と呼ばれる大規模言語モデルの一種。OpenAI 社は 2022 年 11 月に GPT-3.5 を利用した「ChatGPT-3.5」を無料で公開し、2023 年 3 月には GPT-4.0 を利用した「ChatGPT-4.0」を有料で公開している。ChatGPT-4.0 は、3.5 よりも高精度で文書作成だけでなく画像・音楽・動画の生成も可能。

2.4 入力と出力の流れ

(1)　三脚に固定した Web カメラからフローリングワイパーの色をトラッキングする.

(2)　トラッキングした座標を Unity に送る

(3)　受け取った座標にオブジェクト（マレット）を追従させる.

(4)　プロジェクターから Unity 内の画面の床に出力させる.

2.5　カラートラッキング

2.5.1 カラートラッキングについて

カラートラッキングは Python によって動作している.OpenCV のライブラリを使用しカメラから読み取った映像から HSV 値で色を読み取る.HSV 値とは色相（Hue）,彩度（Saturation）,明度（Value）の 3 つの成分で色を表現する方法.その読み取った値から座標を取得し Unity に送り,リアルタイムでオブジェクトを追従させる.

2.5.2　HSV 値設定

カラートラッキングは cvzone のライブラリを使用し取得したい HSV 値を設定する.
下の値は赤色の HSV 値。

main.py

```
hsvVals = {‘hmin’:0, ‘smin’:100, ’vmin’:0, ‘hmax’:11, ‘smax’:255, ‘vmax’: 255}
```

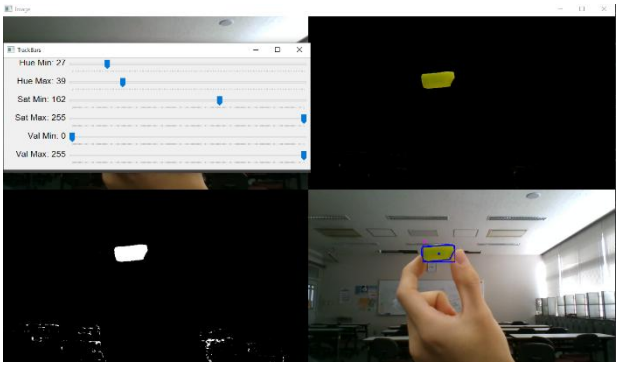


図 4　HSV 値設定後

左上の画面で HSV 値の閾値を変化させる UI。

右上で読み取った色。

左下はマスク。

右下は読み取った範囲を長方形の枠で囲っている。

2.6 カメラ設定

OpenCV には、解像度、明るさ、コントラスト、色相、ホワイトバランスなどを始めとした、カメラの設定を確認、変更するコマンドが用意されている。

カメラの設定を変更するためのコマンドが set（）である。カラートラッキングで使用するカメラは Ｗｅｂカメラであるので、VideoCapture(1)に設定する。(0) はデフォルトのカメラ（ＰＣ内蔵カメラ）である。cap.set(3,1280)とは、ビデオキャプチャのプロパティを設定であり、cap.set(propId, value)の形式で、propId にはプロパティの ID、value には設定したい値を指定する。ここでの 3 はプロパティ ID で、cv2.CAP_PROP_FRAME_WIDTH に対応し、キャプチャするフレームの幅をピクセル単位で指定する。つまり、1280 はフレームの幅を 1280 ピクセルに設定している。cap.set(4,760)とは同様に、ビデオキャプチャのプロパティを設定しているが、4 は cv2.CAP_PROP_FRAME_HEIGHT に対応し、キャプチャするフレームの高さを指定している。760 はフレームの高さを 760 ピクセルに設定してい

る。cap.set(4,760)は同様に、この行はビデオキャプチャのプロパティを設定しますが、4 は cv2.CAP_PROP_FRAME_HEIGHT に対応し、キャプチャするフレームの高さを指定する。760 はフレームの高さを 760 ピクセルに設定している。

```
cap = cv2.VideoCapture(1)
cap.set(3,1280)
cap.set(4,760)
```

図 5　ビデオキャプチャのプロパティ

2.7　Python と Unity の連動

Python と C#とのデータのやり取りをするためにソケット通信を行う.

そのためには,Socket というライブラリを使用し,自分自身を指す IP アドレスとポート番号を指定している.

```
sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
serverAddressPort = ("127.0.0.1",5052)
```

図 6　ソケットの生成

ソケット通信は UDP と TCP があるが,通信速度が UDP のほうが速いため UDP のソケット通信を使用する.

第 3 章　研究概要

3.1 成果物

まず、この研究ではフローリングワイパーに赤い紙を巻きつけ、それをポインタとして読み取って直感的に操作できるホッケーゲームの制作をしました。

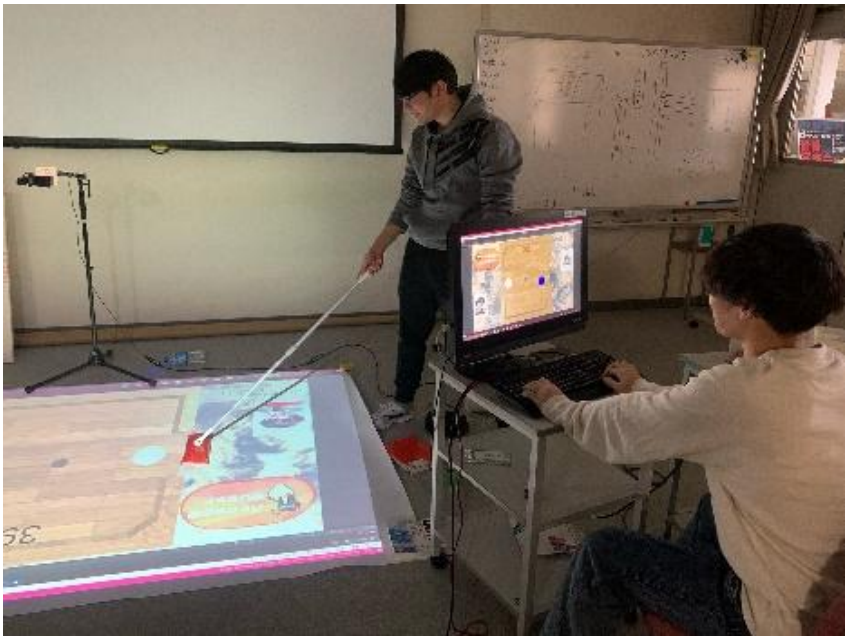


図 7　成果物

3.2 開発環境について

本研究の開発環境を以下の表 1 に示す.

OS	Windows10
開発環境	Unity,PyCharm
主なライブラリ	OpenCV, cvzone
使用言語	C# , Python

表 1　開発環境

3.3 仕様技術の紹介

3.3.1　Unity

Unity は、Unity Technologies によって開発されたマルチプラットフォームゲームエンジンであり、2D、3D、仮想現実（VR）、拡張現実（AR）を含む多様なゲーム及びインタラクティブコンテンツの制作を可能にする。Unity のエディタは、シーン構築からアセット管理、アニメーション作成まで、開発プロセスの全般をサポートし、アセットの利用により、プロジェクトの迅速化を促進する。その多用途性でゲーム開発を超えた分野においても採用されている。

3.3.2 PyCharm

PyCharm は、Python に特化した IDE の一つで、業務の効率化を図りたい Web 開発者やデータサイエンティストなどの間で多く活用されている。Python を開発する際の IDE の定番ともいわれる PyCharm には、プログラミング言語を学習できる機能がついたエディションもあり、初心者の学習や本格的な開発など幅広く利用できるのが特徴である。

3.3.3 C#

C#は,マイクロソフトが開発したプログラミング言語である.そのためオブジェクト指向が取り入れられている.また,C#はプログラミング言語の中でも比較的使いやすく,初心者にとって易しいのが特徴であり,「Unity」と相性が良いことで有名である.

3.3.4 Python

Python(パイソン)はインタープリタ型の高水準汎用プログラミング言語である.Python（パイソン）はオープンソースのプログラミング言語の一つで、1991 年に開発されたインタープリタ型の高水準汎用プログラミング言語である。シンプルで読みやすい構文をもち、数値計算から Web アプリ開発、AI 開発など幅広い用途で利用できることが特徴である。また、豊富なライブラリやフレームワークが多く活用しやすいことや、オープンソースで大規模な開発者コミュニティが存在していることから、初心者から専門家まで幅広い層に人気がある。

3.3.5 OpenCV

OpenCV とは、Intel が開発した画像処理・画像解析のためのオープンソースのライブラリである。正式名称は、「Open Source Computer Vision Library(オープン・ソース・コンピュータ・ビジョン・ライブラリ)」だ。この OpenCV は、誰でも無料で利用することができ、利用用途も趣味や娯楽から商用利用まで幅広く対応している。さらに、プログラミングの知識があまりない方でも気軽に利用できることから、世界中で多くのユーザーに利用されている.

3.3.6 cvzone

cvzone は、画像や動画を処理するためのプログラムを簡単に作れるようにする Python のライブラリである。例えば、人の顔を認識したり、動いている物を追いかけたり特定の色を見つけ出したりすることが可能である。このライブラリは、プログラミングでよく使われる OpenCV というツールと組み合わせて使われることが多く複雑な操作を簡単に実行できるように設計されている。cvzone を使用することで、少ないコードで高度な画像処理が可能になり、研究や開発、趣味のプロジェクトで幅広く活用できる。今回はカラートラッキングで使用する。

第４章　スタートシーン

4.1　スタートシーンについて

本ゲームのシーンについての説明とその中でどのような処理が行われているのか、そしてＣ＃によるシステムの制御方法について記す。

ゲームが始まり一番初めに表示されるシーン「**スタートシーン**」について説明していく。

図８　スタートシーン

4.2　テキストアニメーション

このシーンではタイトルテキストが下記のスクリプトに常に上下している。

```
private void UpdateAnimation()
{
    tmpText.ForceMeshUpdate(true);
    tmpInfo = tmpText.textInfo;

    var count = Mathf.Min(tmpInfo.characterCount, tmpInfo.characterInfo.Length);
    for (int i = 0; i < count; i++)
    {
        var charInfo = tmpInfo.characterInfo[i];
        if (!charInfo.isVisible)
            continue;

        int matIndex = charInfo.materialReferenceIndex;
        int vertIndex = charInfo.vertexIndex;

        Vector3[] verts = tmpInfo.meshInfo[matIndex].vertices;

        float ofs = 0.5f * i;
        float sinWave = Mathf.Sin((ofs + Time.realtimeSinceStartup * Mathf.PI * speed) / length) * amp;
        verts[vertIndex + 0].y += sinWave;
        verts[vertIndex + 1].y += sinWave;
        verts[vertIndex + 2].y += sinWave;
        verts[vertIndex + 3].y += sinWave;
    }

    for (int i = 0; i < tmpInfo.materialCount; i++)
    {
        if (tmpInfo.meshInfo[i].mesh == null) { continue; }

        tmpInfo.meshInfo[i].mesh.vertices = tmpInfo.meshInfo[i].vertices;
        tmpText.UpdateGeometry(tmpInfo.meshInfo[i].mesh, i);
    }
}
```

図 9　テキストアニメーション

このスクリプトを使用しテキストにアニメーションを付けるためにはいくつかの手順が必要になる。以下に画像とともに手順を示していく。

1．動かしたいテキストを選択する

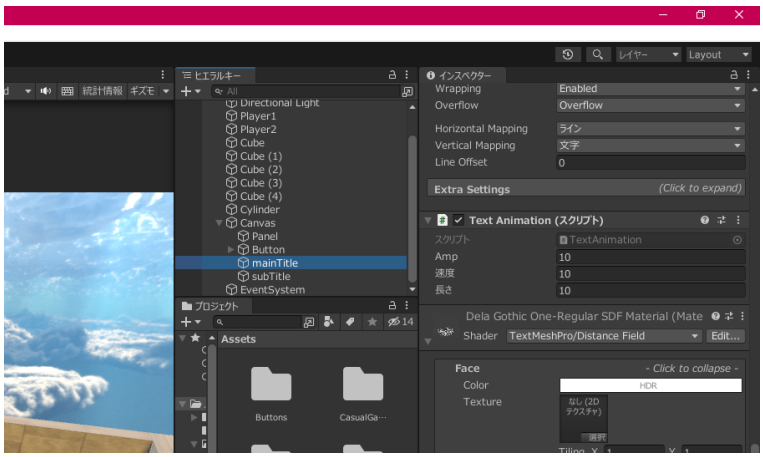


図 10　　テキストの選択

今回の場合は mainTitle を動かしたいため上図のように右側に mainTitle のインスペクター画面が表示されていればよい。

2. インスペクター画面にカギを付ける

画面右上に小さい鍵マークがあるのが分かる。鍵の部分をクリックすると鍵を付けることができる。鍵を付けると別のオブジェクトを選択してもインスペクター画面が変移せず常に同じインスペクター画面を開き続ける。

図 11　インスペクター画面の固定

3.スクリプトのアタッチ

「コンポーネントを追加」へ TextAnimation スクリプトをドラッグアンドドロップする

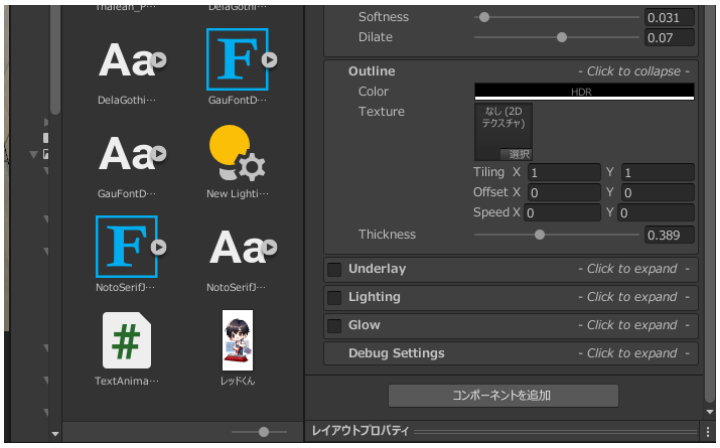


図 12　スクリプトのアタッチ

これでタイトルにアニメーションを付けて動かすことができるようになる。

4.3　ボタンによる画面遷移

次にボタンによる画面遷移の仕方を解説していく。

1. パネル，ボタンの配置

図 13　パネル

図 13 のようにゲームオブジェクト→UI→パネルの順で追加する。

図 14　ボタン

次にパネルの階層の下にボタンを配置する。
こうすることによって UI を切り替えるときにまとめて変更することができる。

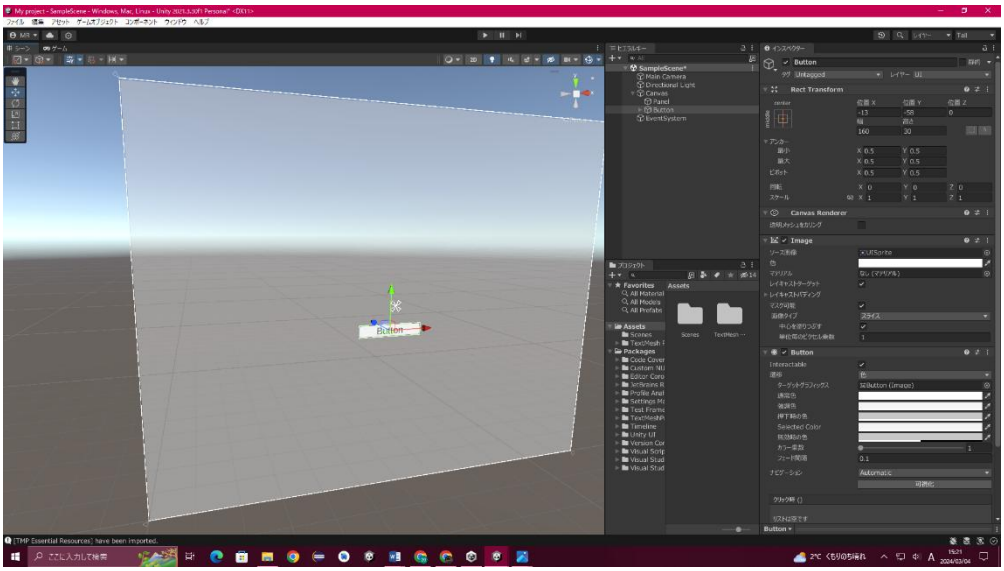


図 15　ボタンの配置

ボタンの配置後のプレビュー画面である。

2. スクリプトのアタッチ

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class ExplainStart : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {
        this.GetComponent<Button>().onClick.AddListener(push);
    }

    // Update is called once per frame
    void push()
    {
        SceneManager.LoadScene("ExplainScene");
    }
}
```

図 16　画面遷移のスクリプト

上記のコードが画面遷移時に実行されるスクリプトである。

このスクリプトをボタンにアタッチするとボタンによる画面遷移が可能となる。

第 5 章　ルール説明シーン

スタートシーンのスタートボタンをクリックすることで、画面が遷移し、ルール説明シーンが表示される。

図 17　ルール説明シーン

5.1 シーンの構成と UI 設定

ルール説明シーンは ExplainPanel に入っているテキスト UI やボタン UI で構成されている。

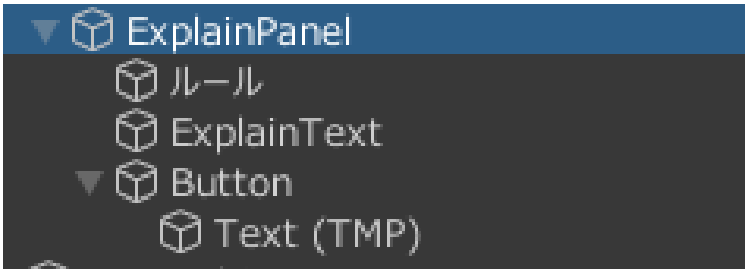


図 18　ルール説明シーンの UI 構成

以下は画面上部の「ルール」という文字を表示する UI の設定である。

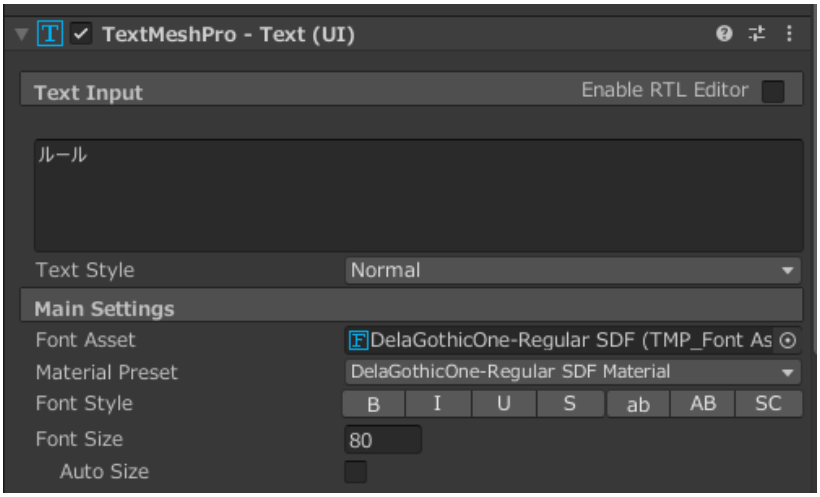


図 19　ルールテキストの UI 設定

「Text Input」という設定に表示したいテキストを書くことで表示できる。

TextMeshPro の UI を使用して表示している。

以下がルールの本文の設定である。

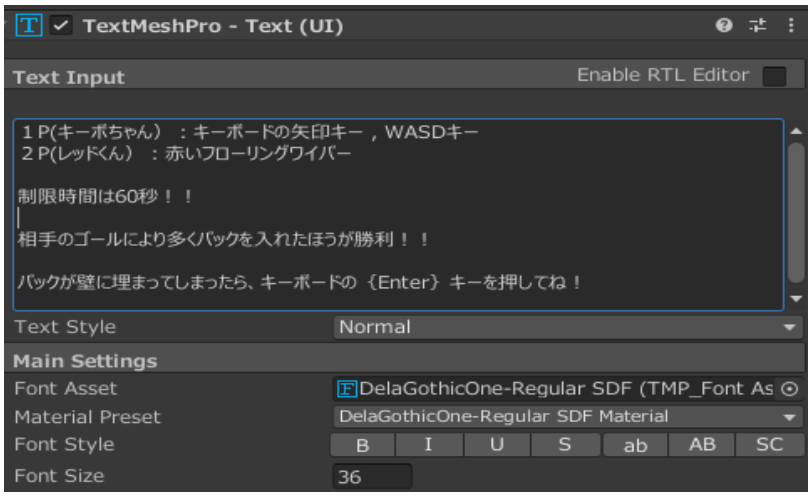


図 20　ルールテキスト本文の UI 設定

以下がボタンの設定である。

図 21　ボタン UI の設定

Game Start というスクリプトにボタンを押したときの処理が書かれているため、Game Start スクリプトをアタッチしている。以下がボタンのテキストの設定である。

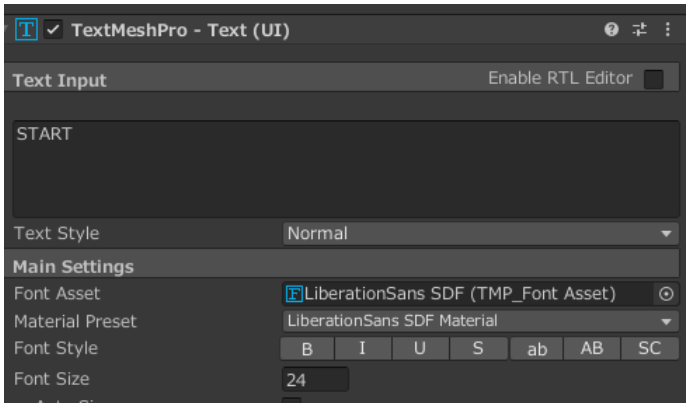


図 22　ボタンのテキスト UI の設定

5.2 Text と TextMeshPro の違い

Text と TextMeshPro はどちらも Unity 内にテキストを表示させる UI であるが、仕様が少し異なる。TextMeshPro を使用すると、Text に比べてテキストのフォーマットとレイアウトに際してより高度な制御が可能になる。しかし、TextMeshPro はデフォルトでは英数字しか使用できないため、日本語で使用したい場合はフォントアセットを作成する必要がある。また、デフォルトのテキストのカラーは白である。

一方、Text は text コンポーネントでそのまま文字を設定することができるため、直感的に簡単に使えるという魅力がある。

5.3 テキストの表示

TextMeshPro ではデフォルトで英数字しか表示できないため、フォントアセットを使用し、日本語で表示できるようにした。

フォントは GoogleFont からダウンロード

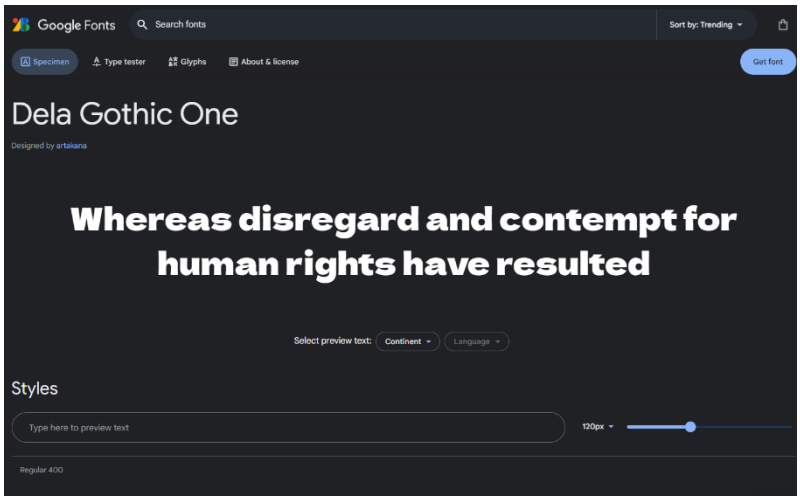


図 23　Dela Gothic One のダウンロードページ

フォントダウンロード後、ウィンドウ ＞ TextMeshPro＞ フォントアセットクリエーター

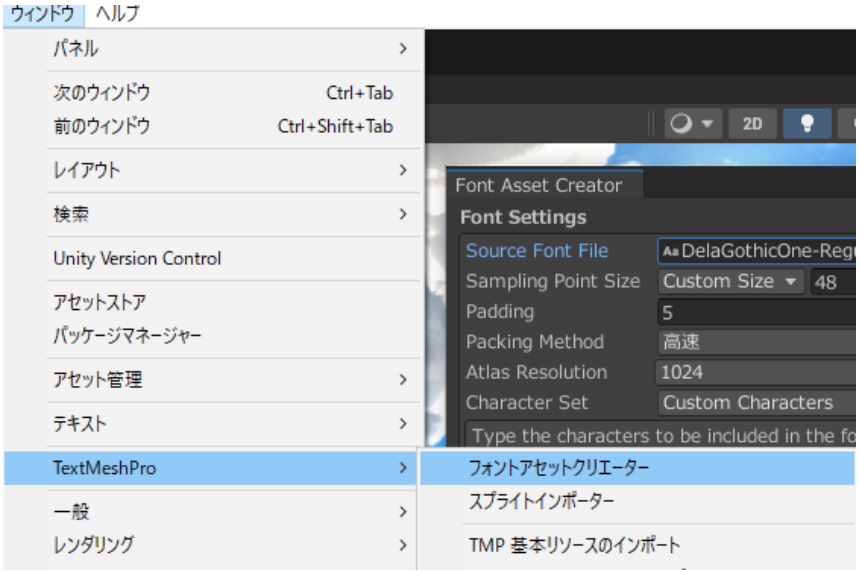


図 24　フォントアセットクリエーター

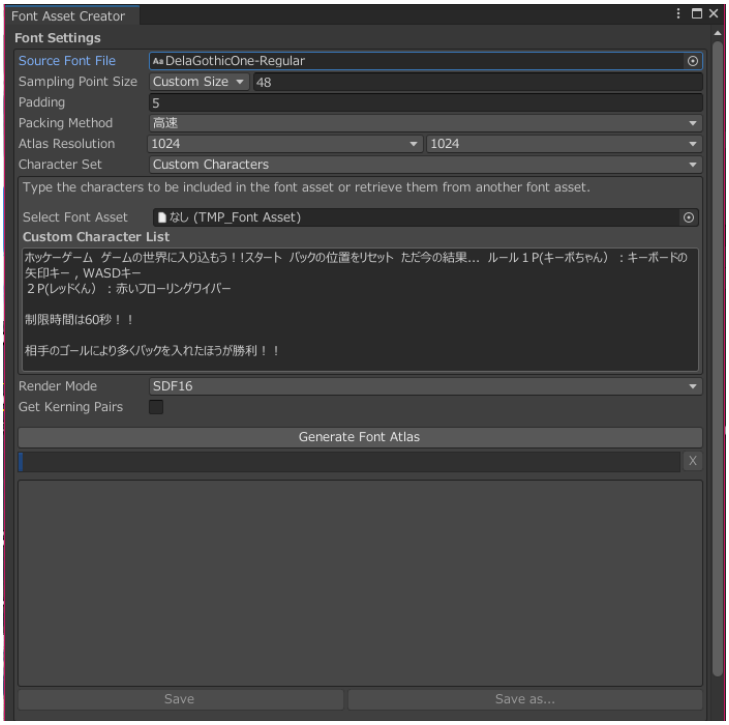


図 25　フォント設定

Source Font File でダウンロードしたフォントを選択し、設定できる。

Custom Character List に使いたい文字を書き込み Generate Font Atlas をクリックし、Save をクリックすると使用できるようになる。

5.4 背景の透過処理

ExplainPanel を透過させるにはインスペクター画面から Image の色を選択する。

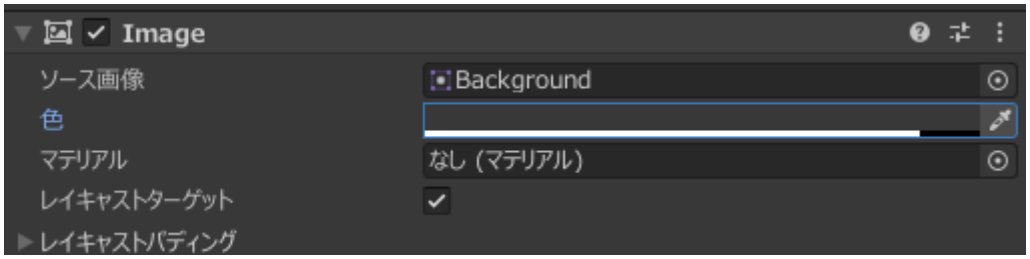


図 26　ExplainPanel の色を選択

その後、RGBA の A の値を小さくすればパネルの透過ができる。

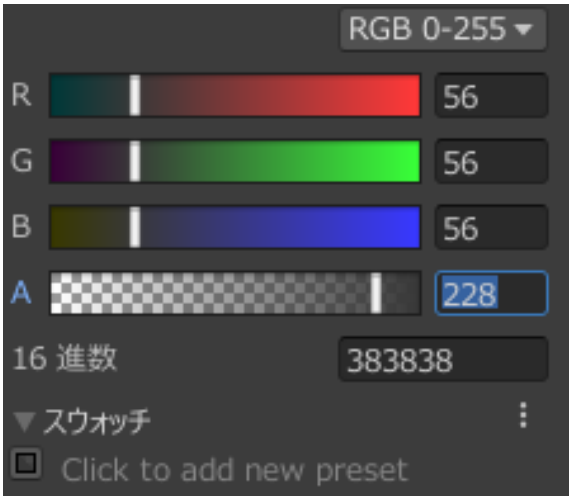


図 27　透過処理

5.5 ボタンの処理

ボタンの処理はスタートシーンと同様

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class ExplainStart : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {
        this.GetComponent<Button>().onClick.AddListener(push);
    }

    // Update is called once per frame
    void push()
    {
        SceneManager.LoadScene("ExplainScene");
    }
}
```

図 28　ボタンの処理のコード

第 6 章　ゲームシーン

左側にレッドくん、右側にキーボちゃんの画像を表示している。

図 29　ゲームシーン画面

6.1 カウントダウン・タイマーについて

ルール説明画面から画面が遷移したら、「3，2，1，GO!」というカウントダウン（以後、カウントダウンと呼ぶ）がスタートし、その後 60 秒タイマー（以後、タイマーと呼ぶ）がスタートする。このふたつの処理の流れを GameTimer という関数で処理している。カウントダウン及びタイマーの処理は Time.deltaTime というプロパティを使い処理している。Time.deltaTime とは直前のフレームと今のフレーム間で経過した秒を返すプロパティである。Countdown という変数に初期値 3 を設定し、そこから Time.deltaTime で取得した秒数を引いていくことでカウントダウンを表現している。

以下は、カウントダウンの処理である。

```
void DoCountdown()
{
    countdown -= Time.deltaTime;
    if (countdown > -1)
    {
        CountdownText.text = "3";
    }
    else if (countdown > -2)
    {
        CountdownText.text = "2";
    }
    else if (countdown > -3)
    {
        CountdownText.text = "1";
    }
    else if (countdown > -4) // countdownが-4以下になったら「GO!」を表示
    {
        CountdownText.text = "GO!";
    }
    else if (countdown > -5) // countdownが-5以下になったらゲーム開始
    {
        if (!gameStarted) // ゲームがまだ開始していない場合のみStartGameを呼び出す
        {
            StartGame();
        }
    }
}
```

図 30　カウントダウンの処理

StartGame()に入ったら、gameStarted のフラグを True にしゲームを開始させる。

カウントダウンを非表示にするには、カウントダウンのテキストのゲームオブジェクトの設定（Set）から Active を以下の通りに false に変更する。

タイマーテキストは同様に true に変更し表示させる。

```
void StartGame()
{
    gameStarted = true;
    CountdownText.gameObject.SetActive(false); // カウントダウンテキストを非表示にする
    TimerText.gameObject.SetActive(true); // タイマーテキストを表示にする
}
```

図 31　StartGame()の処理

DoGameTimer()はタイマーの処理である。

Timer の初期値を 60 にし、そこから Time.deltaTime で取得した秒数を引いていくことでタイマーを表現している。タイマーテキストの変数に整数型にキャストした値を文字列に変換して TimerText のテキストプロパティに設定する。そうすることでゲーム画面にタイマーが表示される。

```
Timer -= Time.deltaTime;
TimerText.text = ((int)Timer).ToString();
```

図 32　制限時間を表示

タイマーが 0 以下になった時はフォントサイズを小さくして、見えなくしている。

6.2 判定分岐

タイマーの処理が終わったら CheckWinner()という関数が動作する。

6.2.1　ゴールの判定

ゴールの位置に透明なオブジェクトがあり、そこにパックが衝突すると点数が加算される。透明なオブジェクトはレッドくん側とキーボちゃん側にそれぞれあり、レッド君側の透明なオブジェクトにパックが衝突するとキーボちゃんの得点。キーボちゃん側の透明なオブジェクトにパックが衝突するとレッドくん側の得点となるように GoalManager.cs というスクリプトを書いている。

点数表示をリアルタイムで表示を変更させるために以下の通りにプログラムを書いている。

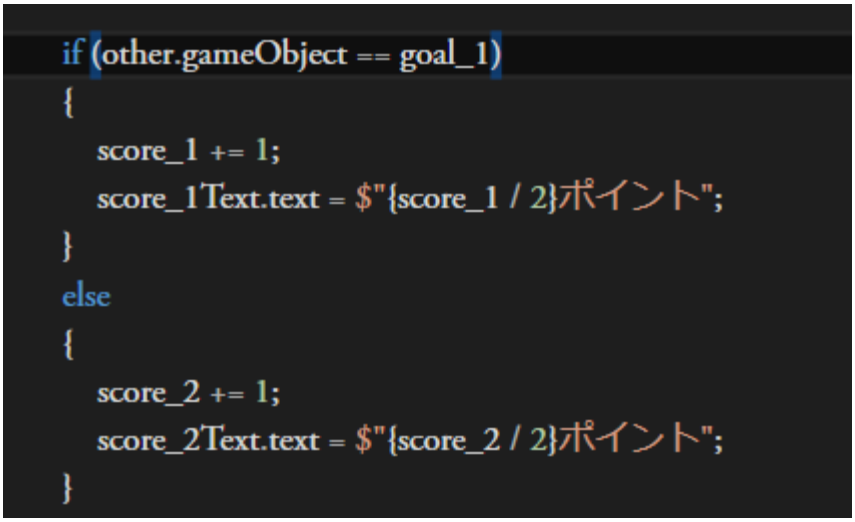


図 33　ゴールの判定

得点を 1/2 にしている理由は、ゴールオブジェクトにパックが衝突したときに 2 回衝突したことになっていたため、1/2 にすることで正常に点数が 1 ずつ加算されるようにした。

6.2.2　勝敗判定の分岐

タイマーが 0 になった時の両者の得点を比較して、画面に表示するテキストを分岐させる。goalManager.score_1 がレッドくんのスコアで、goalManager.score_2 がキーボちゃんのスコアである。

```
void CheckWinner()
{
    if (goalManager.score_1 > goalManager.score_2)
    {
        WinnerPlayer1Window.SetActive(true);
    }
    else if (goalManager.score_2 > goalManager.score_1)
    {
        WinnerPlayer2Window.SetActive(true);
    }
    else
    {
        WinnerDrawWindow.SetActive(true);
    }
    // 勝者が決まった後はClearWindowを非表示にする
    ClearWindow.SetActive(false);
}
```

図 34　スコアの比較及び分岐

図 35　レッドくんが勝った場合

図 36　キーボちゃんが勝った場合

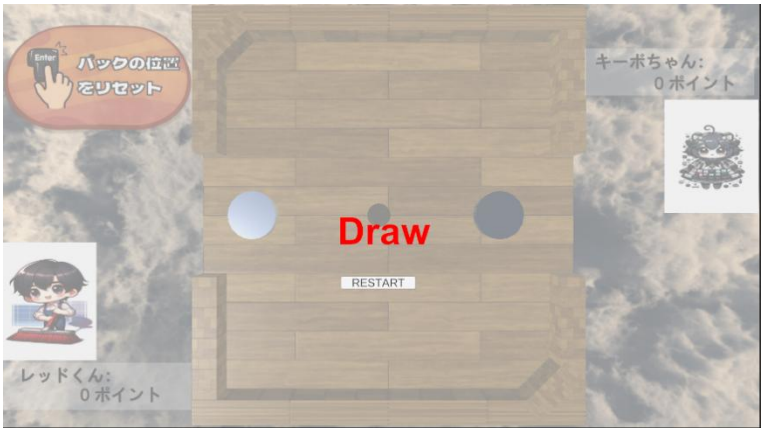


図 37　同点だった場合

6.2.3　True False による表示 UI の切り替え

UI の表示はインスペクター画面のチェックマークで管理ができる。

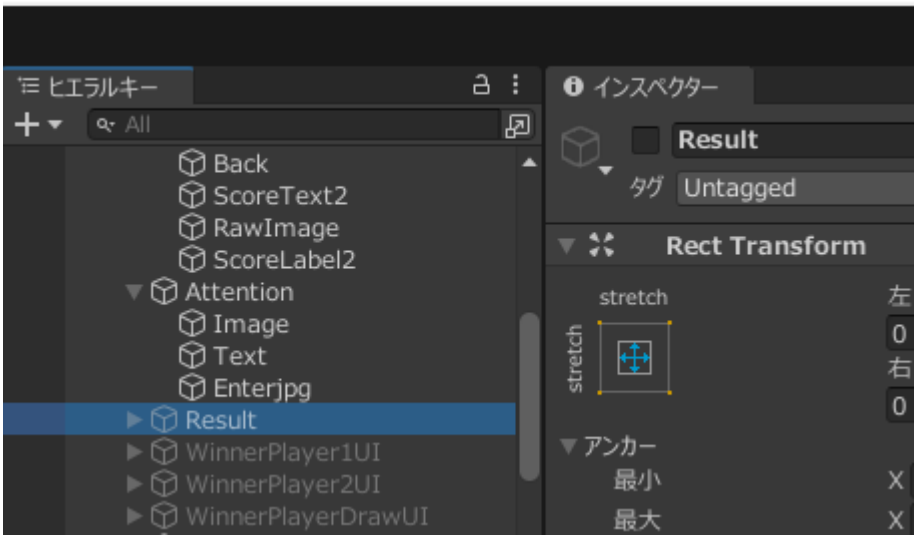


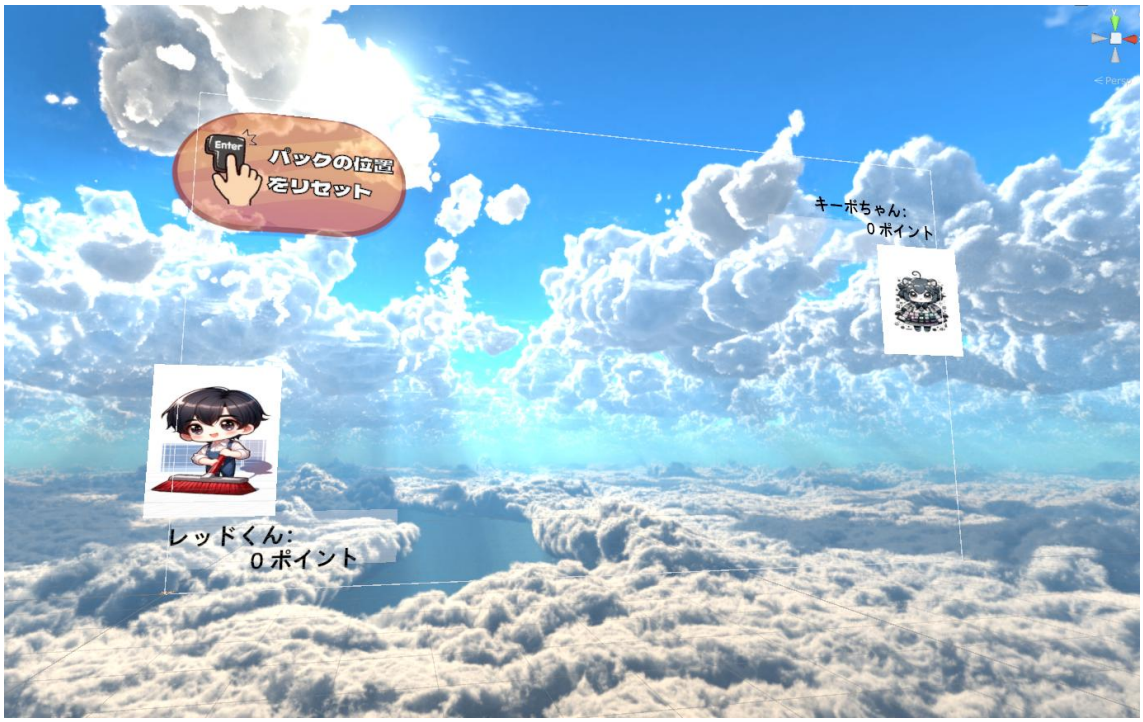
図 38　非表示 UI

上の画像ではチェックマークがついておらず Result という UI オブジェクトの文字も半透明。この状態だと UI が表示されていない。下の画像では UI が表示されている状態。チェックマークがついており Result という UI オブジェクトの文字も白く表示されていることが分かる。

図 39　UI の表示

6.3　UI とステージについて

6.3.1　プレイヤー画像、得点の配置



プレイヤー画像については得点とセットにして表示するときにわかりやすいように配置。

6.3.2　Enter キーによるパックの位置のリセット

図 39　Enter キーUI

Enter キーを押すとパックの位置をリセットできるということが分かりやすいように Enter キーの画像と共に UI を設置した。

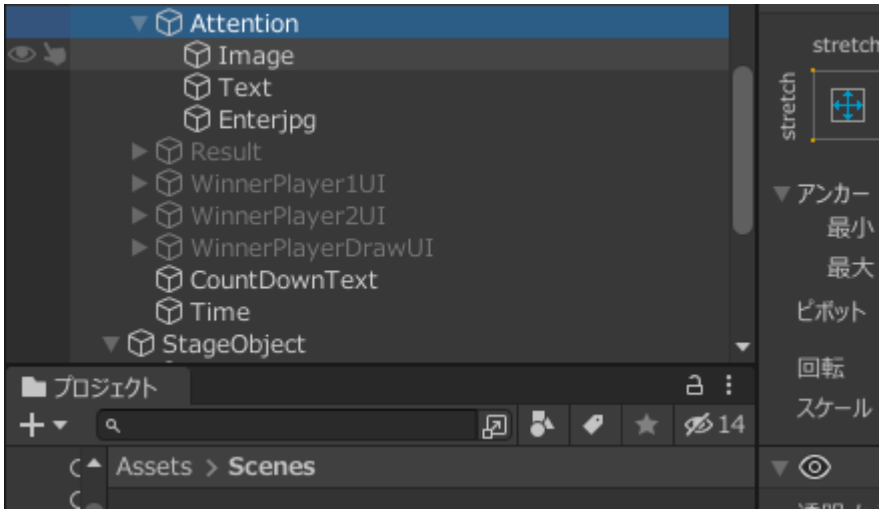


図 40　Enter キーUI の構造

上記の図が Enter キーUI の画像の設置である。

Attention がパネルの役割を果たす。その中に茶色い楕円形の背景、「パックの位置をリセット」というテキスト、Enter キーの画像を配置している。

6.3.4　キーボード操作の移動制限

キーボード側の操作で、パックを壁に押し込むことによってパックが壁に埋まってしまう問題が発生した。その問題を解決するため、マレットに移動制限を付けパックを壁に押し込むことができないようにした。

```
void Start()
{
    playerRB = this.gameObject.GetComponent<Rigidbody>();
}

void Update()
{
    // Enterキーが押されたかどうかをチェック
    if (Input.GetKeyDown(KeyCode.Return))
    {
        GameObject.Find("Cylinder").transform.position = new Vector3(1.26f, 0.6f, 6.09f);
        playerRB.velocity = Vector3.zero;
        playerRB.angularVelocity = Vector3.zero;
    }
}
```

図 41　Enter キーによる位置リセットスクリプト

それでもパックが壁に埋まってしまいゲームの続行が不可能になったときは上記のコードにより、Enter を押されたときにパックの位置をリセットさせることで、ゲームの進行を可能とする。Start 関数でオブジェクトの初期座標を保存し、Update 関数で保存された座標を使用し、オブジェクトの移動を瞬時に可能とする。

6.3.5　キーボード操作

```
public class MoveArrow : MonoBehaviour
{
    public float speed = 5.0f; // オブジェクトの移動速度
    public float minX = -4f;
    public float maxX = -4f;
    public float minZ = -6.5f;
    public float maxZ = 6.5f;

    void Update()
    {
        float moveHorizontal = Input.GetAxis("Horizontal");
        float moveVertical = Input.GetAxis("Vertical");

        Vector3 movement = new Vector3(moveHorizontal, 0.0f, moveVertical);

        // 90度回転させる
        Quaternion rotation = Quaternion.Euler(0, -90, 0); // Y軸を中心に90度回転
        movement = rotation * movement;

        movement *= speed * Time.deltaTime;
        Vector3 newPosition = transform.position + movement;

        newPosition.x = Mathf.Clamp(newPosition.x, minX, maxX);
        newPosition.z = Mathf.Clamp(newPosition.z, minZ, maxZ);

        transform.position = newPosition;
    }
}
```

図 42　キー移動スクリプト

movement *= speed * Time.deltaTime;:は、移動速度を調整し、フレームレートに依存しないように Time.deltaTime を掛け合わせる。これにより、どんなフレームレートでも一定の速度で移動する。

newPosition は、現在の位置に移動ベクトルを加えた新しい位置である。

6.3.6　オブジェクトの配置

ステージを綺麗な四角形にしてしまうと、ゲームをプレイ中にパックが左端と右端に沿うように移動した場合マレットに当たっても跳ね返らずにパックが取れなくなってしまいゲームがスムーズに行えない。

そのため、ステージの角に斜めの角度をつけることによって端に沿うように移動したパックにも動きが生まれゲームをスムーズに行えるようになる。

図 43　ステージの工夫

6.4　衝突判定　Rigid body

6.4.1　Rigid body とは

Rigid body を使うと、ゲームオブジェクトを物理特性によって制御することができるようになる。本研究では、リアルなオブジェクトの衝突運動を行わせるため Rigid body をオブジェクトにアタッチしていく。

6.4.2　Rigid body の追加

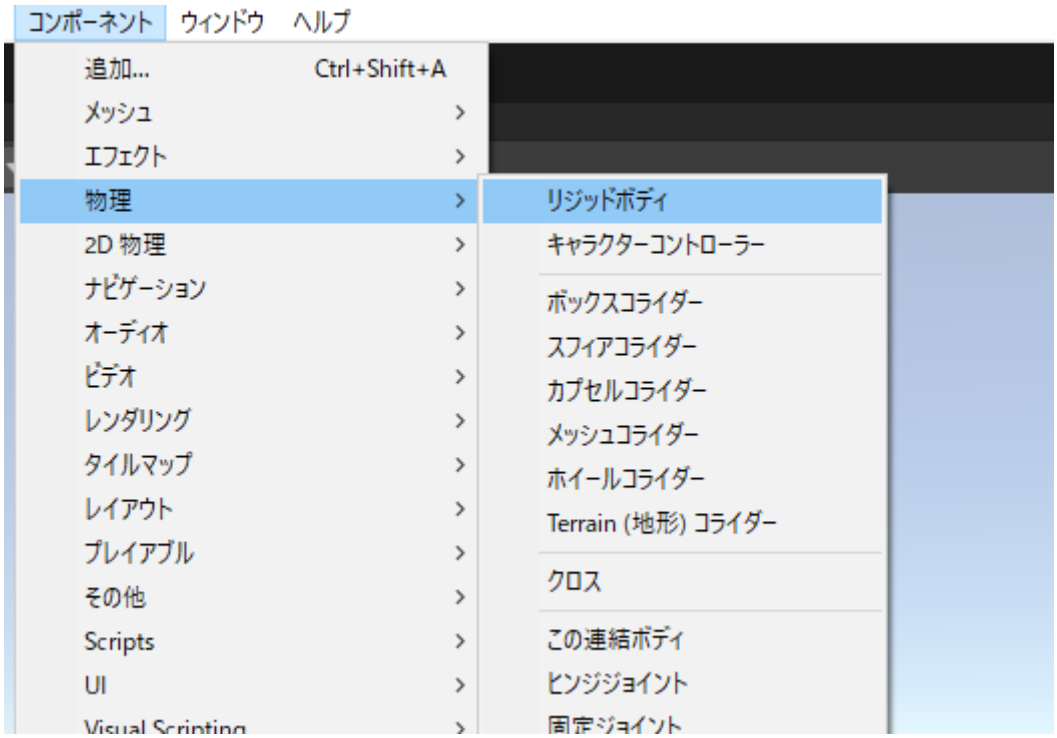


図 44　Rigidbody のアタッチ

アタッチしたいオブジェクトを選び、コンポーネント→物理→リジットボディと選んでいくとリジッドボディを追加できる。

6.4.3　Rigid body の値の変更

Rigid body の値が初期正体のままだと質量が軽すぎるため、スポンジとスポンジが当たったような挙動になる。そのため質量の部分を変更し、本来想定していた動きに近づける。

図 45　質量の値変更

6.4.4　オブジェクトの衝突処理

```
void OnCollisionEnter(Collision collision)
{
    // 衝突点の法線方向に力を加える
    Vector3 forceDirection = collision.contacts[0].normal;
    Vector3 newVelocity = forceDirection * bounceForce + rb.velocity;

    // 新しい速度が最大速度を超えないように制限
    newVelocity = Vector3.ClampMagnitude(newVelocity, maxVelocity);

    // 新しい速度を適用
    rb.velocity = newVelocity;

    audioSource.Play();
}
```

図 46　衝突処理

collision.contacts[0].normal: 衝突した接点の情報の中から、最初の接点の法線ベクトルを取得する。法線ベクトルは衝突面に垂直な方向を示し、この方向に力を加えることでオブジェクトが垂直方向に動く。

Vector3.ClampMagnitude(newVelocity, maxVelocity): 新しい速度の大きさが maxVelocity を超えないように制限する。これにより、オブジェクトの速度が無制限に増加することを防ぐ。

rb.velocity = newVelocity;: 計算された新しい速度をオブジェクトの速度として設定する。rb は Rigidbody コンポーネントを指し、Unity の物理エンジンによってオブジェクトの動きが管理される。

第 7 章　使用したアセット・BGM・フォント

7.1　使用したアセット

使用アセット・BGM・フォントは全て Asset Store、効果音ラボからダウンロードしたもの

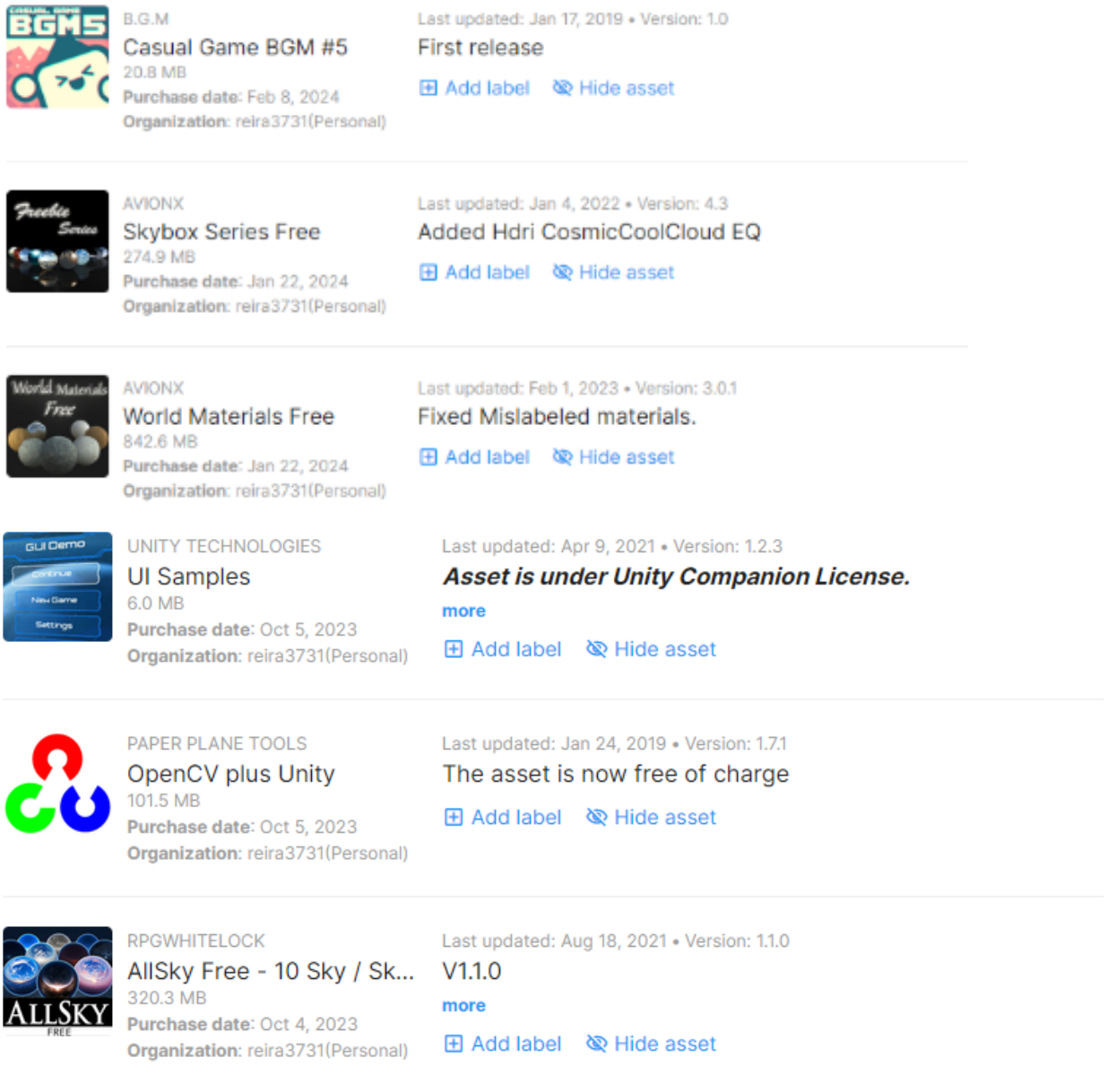


図 47　使用アセット

7.2　BGM の設定の仕方

本研究では、スタート時とゲーム終了時の二つのシーンでそれぞれ一曲ずつ使用している。画像と共に BGM の流し方を解説していく。

1. オーディオソースを追加する。

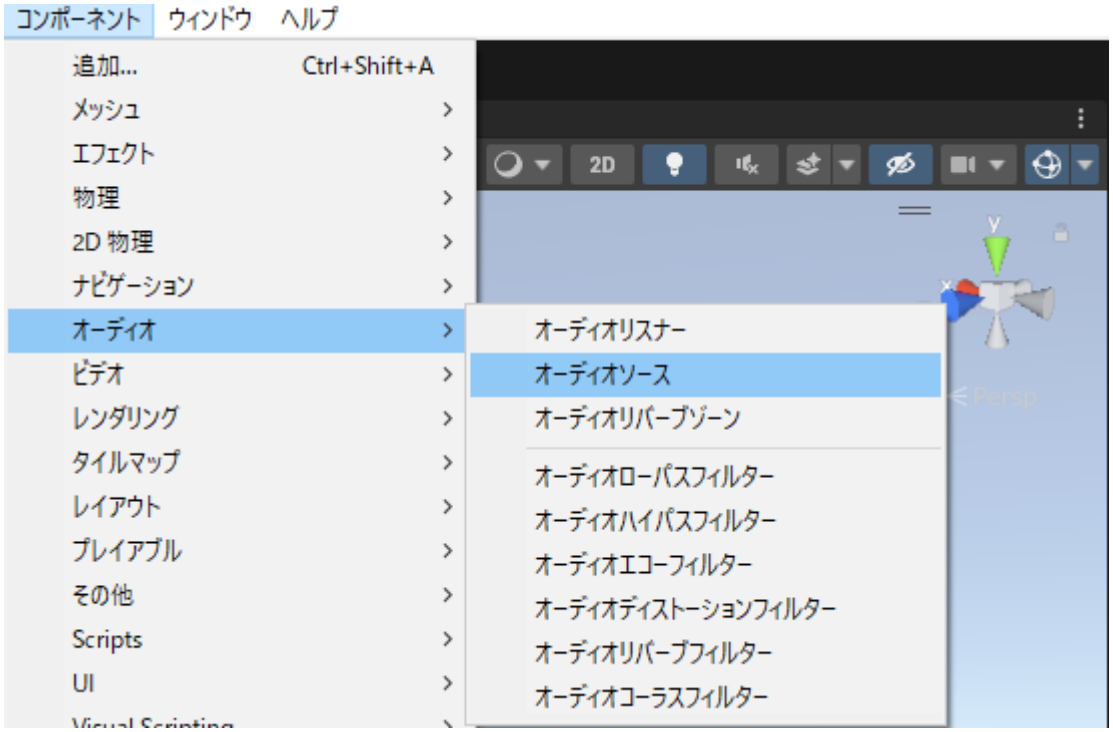


図 48　オーディオソースの設定

コンポーネント→オーディオ→オーディオソースの順に選択し追加する。

2. オーディオソースにダウンロードしてきた音源を追加。

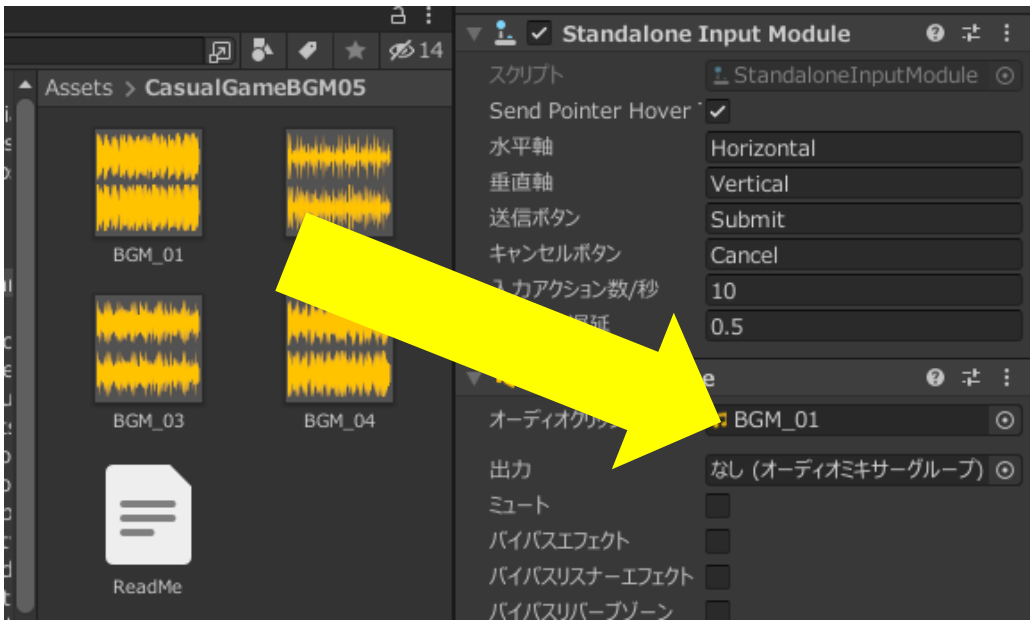


図 49　BGM のアタッチ

上図のようにドラッグアンドドロップ。

3. オーディオソースの設定

図 50　オーディオソースの設定

ゲーム開始と同時に再生したいため「ゲーム開始時に再生」にチェックマークを付ける。

第 8 章　環境構築

8.1 Unity のインストール

一般的には Unity の公式サイトからインストールし、サインインを行えば Unity を使用することができるが,校内プロキシによって,制限されてしまう.そのため,起動用のバッチファイルを作成し,UnityHub を起動する.

```
@echo off

set HTTP_PROXY=http://172.16.0.150:8080

set HTTPS_PROXY=http://172.16.0.150:8080

start "" "C:¥Program Files¥Unity Hub¥Unity Hub.exe"
```

図 51　bat ファイル

このバッチファイルでプロキシを回避して、サインインが行える。

8.2 環境変数の追加

Windows の設定 ＞ システム ＞ 詳細情報 ＞ システムの詳細設定

表 1　環境変数の追加

変数	値
HTTP_Proxy	172.16.150:8080
HTTPS_Proxy	172.16.150:8080

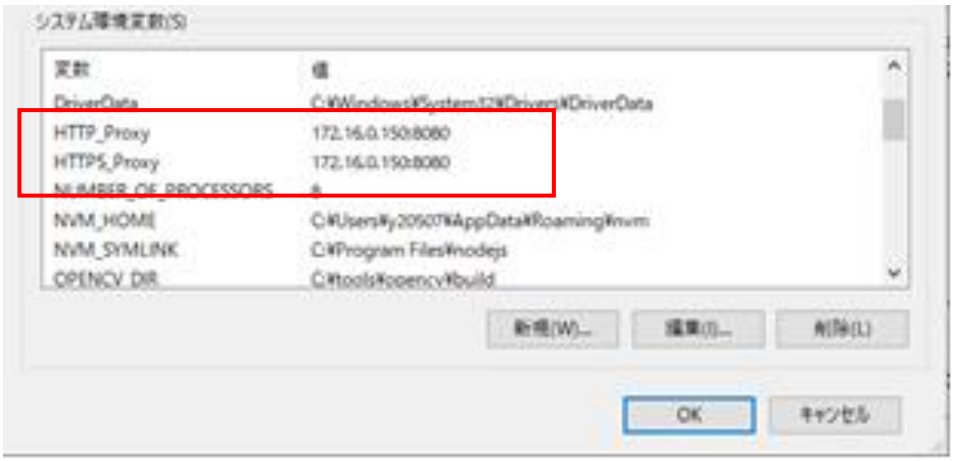


図 52　システムの環境変数

この設定を新規で追加する。

これにより、プロキシを回避できる。

8.3　Python をインストール

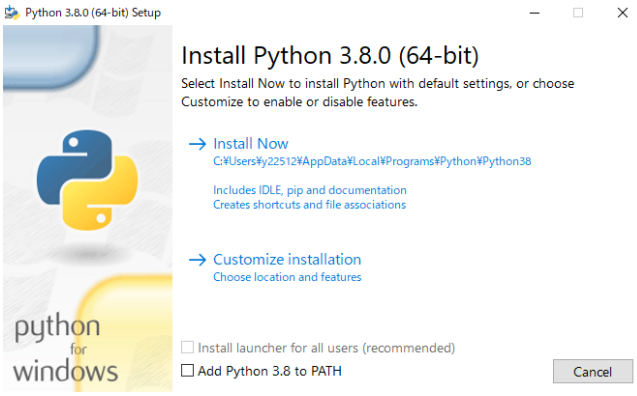


図 53　Python インストール画面

8.4　PyCharm のインストール

PyCharm は,Python に特化した統合開発環境である.無料版を使用する.

公式サイトから「PyCharm Community Edition Setup 」をダウンロードして,セットアップをする.

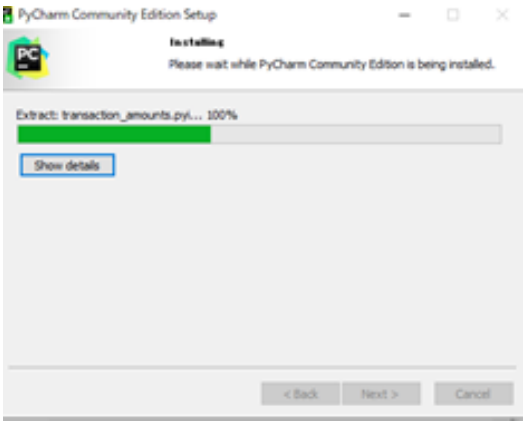


図 54　PyCharm インストール画面

8.5 ライブラリの追加

PyCharm を開く ＞ File ＞ Settings ＞ Add Interpreter
Environment を New にし、Base interpreter を Python 3.8 にし OK を押す。
Python Interpreter に Python3.8 を設定する。

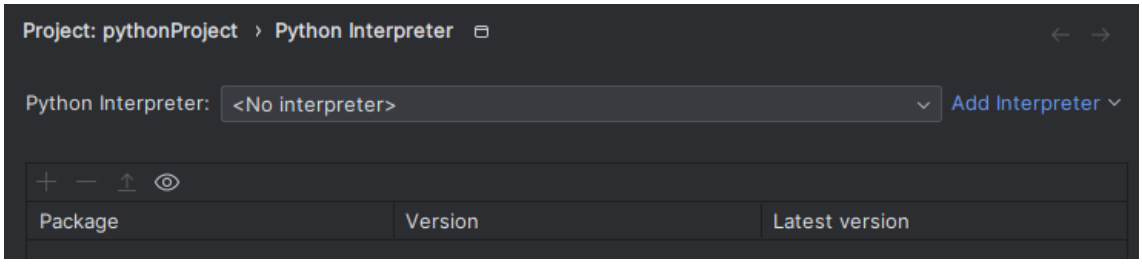


図 55　Interpreter 設定前

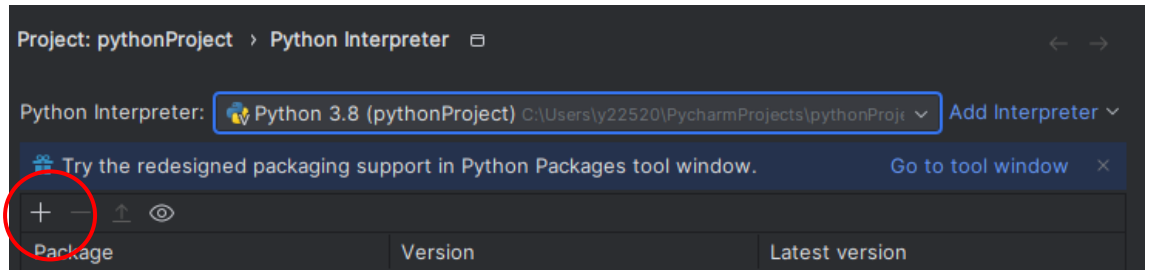


図 55　Interpreter 設定後

赤丸の「＋」をクリックし Available Packages を開き、必要なライブラリをインストールする。

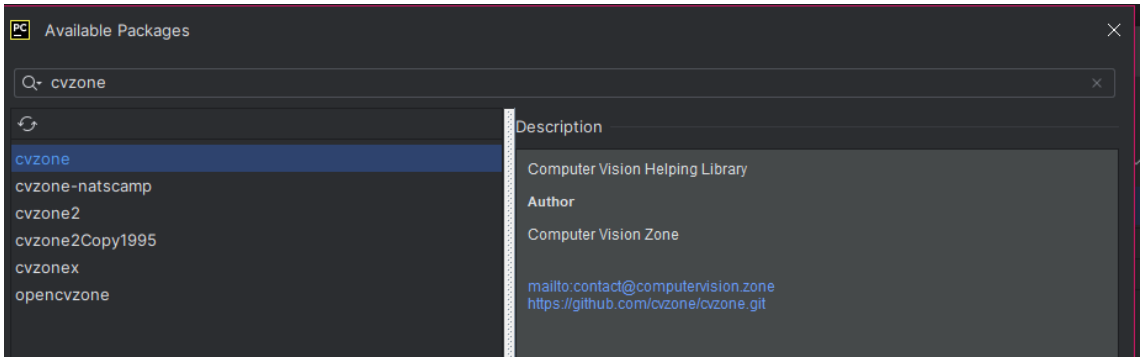
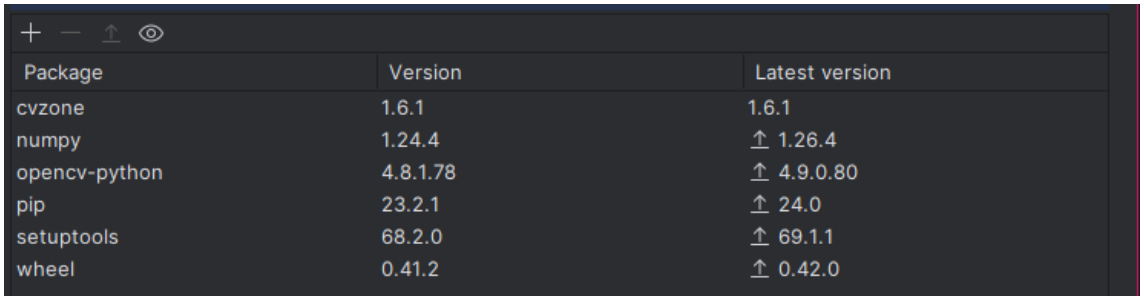


図 56　ライブラリ検索

このように検索すれば出てくるので、インストールする。



Package	Version	Latest version
cvzone	1.6.1	1.6.1
numpy	1.24.4	↑ 1.26.4
opencv-python	4.8.1.78	↑ 4.9.0.80
pip	23.2.1	↑ 24.0
setuptools	68.2.0	↑ 69.1.1
wheel	0.41.2	↑ 0.42.0

図 57　インストールしたライブラリ

第 9 章　ゲーム環境の設備

今回の研究では、実際に体を動かして遊ぶゲームのため、プロジェクターや WEB カメラなど様々な機器を使用する。ここでは、実際に使用した機器や作成物、およびそれらの作成方法について紹介していく。

9.1 プロジェクター

床に映すために木でできた台にプロジェクターを設置する。この台は過去の卒業研究で使われたもの.プロジェクターの設定は通常の画面の向きだと,プレイヤーの背中や,Web カメラの三脚によって,プロジェクターの光が遮られてしまう.そこで,プロジェクターの設定の設置モードをフロント・天吊りに設定し,プレイヤーの位置を画面横にずらすことによって光が遮られずに床に投影できる.また,台形補正機能で投影される画面が長方形になるように設定した.

図 58　プロジェクター台

9.2　WEB カメラ

プレイ中のフローリングワイパーの動きを正確に捉えるためにはフローリングワイパーの動きを真上から撮影しなければならない。そのため、マイクスタンドの先端に、WEB カメラを固定し、真上から動きを検知することにした。

図 59　Web カメラの全体像

9.3　三脚との結合

マイクスタンドを使い Web カメラを固定した.マイクスタンドのネジと Web カメラのネジが違うため直接接続することはできない.そのため,アルミ板を切断し,Web カメラのネジとマイクスタンドのネジそれぞれに合う穴をあけ,マイクスタンドと Web カメラを留め具で結合させた.

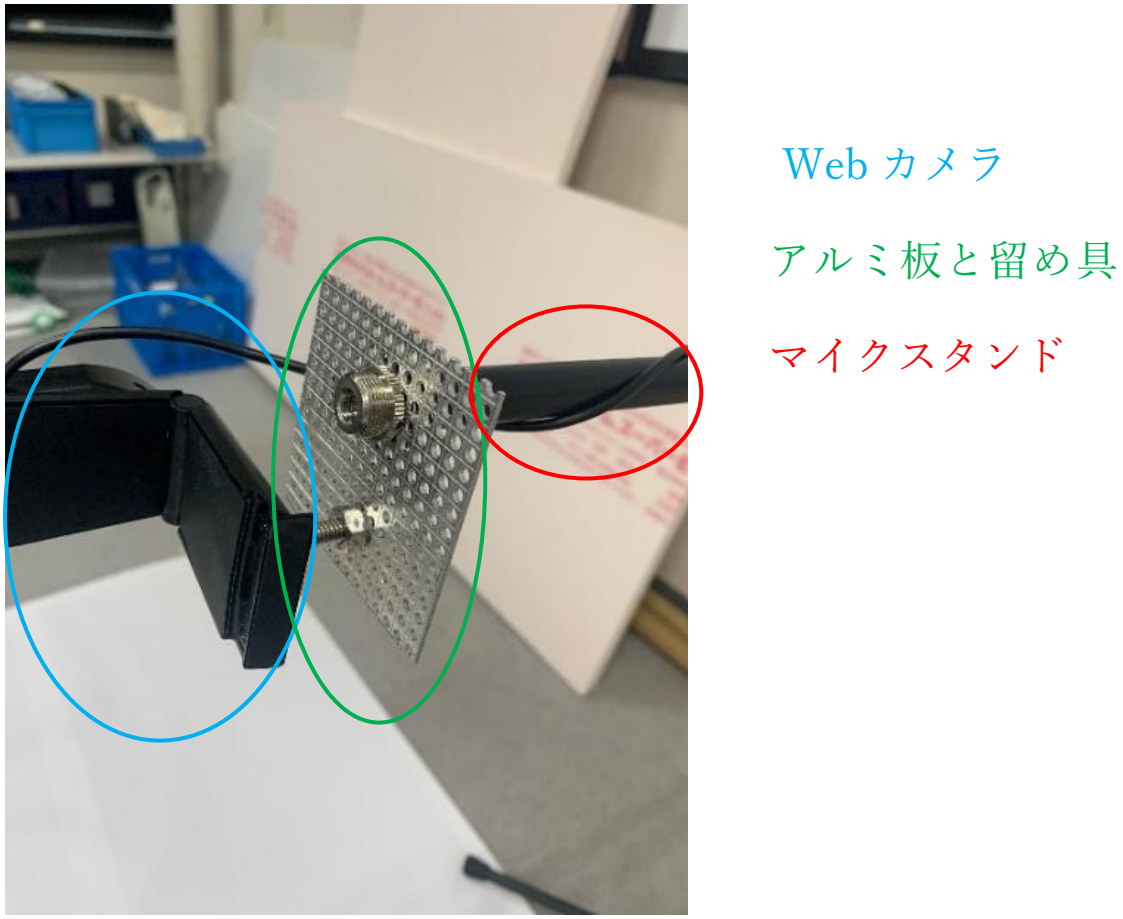


図 60　マイクスタンドと Web カメラの結合部分

9.3.1　ネジ穴のサイズ

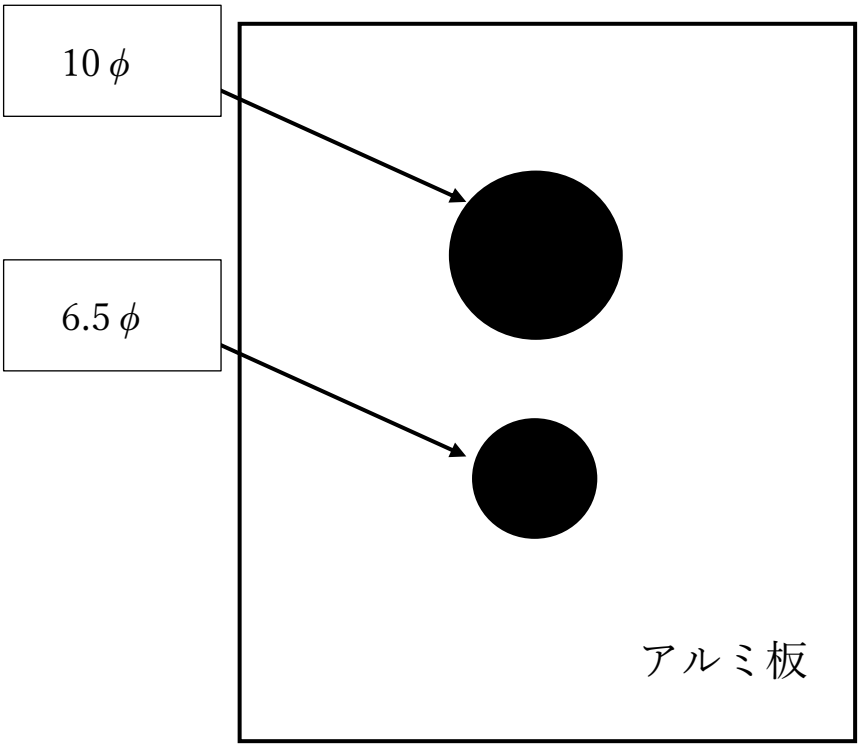


図 61　ネジ穴のサイズ

マイクスタンドのネジ　＝　3／8 インチ　＝　9.525mm　⇒　穴は 10 ϕ

Web カメラのネジ　＝　1／4 インチ　＝　6.35mm　⇒　穴は 6.5 ϕ

9.4　プロジェクタースクリーン

床に映像を投影するので,床が白いほうが綺麗に描画される.また,フローリングワイパーの摩擦が少ないほうがストレスなくゲームをプレイできるので,610mm×幅のインクジェット普通紙を養生テープやのりで貼り,画面のサイズに合わせた一枚のスクリーンを作成した。

9.5 フローリングワイパー

100 均で購入したフローリングワイパーに赤く印刷した紙で包み、テープで張り付ける。これがカラートラッキングのポインタとなる。

図 62　フローリングワイパー

第 10 章 おわりに

研究を通して,Unity や Python の知識が身についた.また,思った通りにプログラムが動かないときの問題解決能力が身についた.本来は,2 人でフローリングワイパーを動かして,プレイできるゲームを作りたかったが,カラートラッキングした座標を 2 つのオブジェクトにアタッチすると,両方ともオブジェクトが動かないという問題が発生し,研究が滞ってしまったため,カラートラッキングで追従させるオブジェクトを 1 つとし,片方のプレイヤーはキーボードで操作するという形式になった.改善策として、カラートラッキングするカメラを 2 台に増やしたり、カラートラッキングのスクリプトをキーボちゃん側とレッドくん側で別々に作ったりすれば床だけでゲームができると考えられる。しかし、私たちの知識不足もありその解決策に気づくのが遅くなってしまったため、来年以降の卒業研究でよりよいものにしてほしい。

結果として,子供だけでなく大人も楽しめるホッケーゲームが制作できた.

第 11 章　参考文献

・斎藤　英比古：Kinect を使った NUI の研究,
平成 23 度産技短卒研報告書,pp6,2012

・【Unity】　文字を表示する UI　TextMeshPro と Text
https://elly-app-creator.blog.jp/archives/19105017.html

・【Python×OpenCV】
カメラの設定を確認・変更する方法（解像度、明るさなど）
https://www.klv.co.jp/corner/python-opencv-camera-setting.html

・【Unity】スクリプトからサウンドの ON/OFF を切り替える方法
https://zakozakocreator.com/unity-sound-onoff-muteproperty/

・【Unity】他のオブジェクトにアタッチされたスクリプトの変数を取得する方法。GetComponent()を使います。
https://game.daraha.me/basic/access/
・カウントダウン後にゲームを開始する方法【Unity 的当て】
https://goodlucknetlife.com/unity-targetgame-countdown-start/

・【Unity ゲーム開発テンプレ】制限時間（カウントダウン）＆タイムアタック（カウントアップ）の実装方法
https://youdoyou-motto.com/control-time-indeveloping

・タイムアップで UI テキストを表示
https://youdoyou-motto.com/control-time-indeveloping

・【unity】位置、姿勢をリセットするボタン作成方法
https://your-3d.com/unity-transform-reset/

・PyCharm（パイチャーム）とは？5 つの特徴や導入する流れ、便利な機能を紹介
https://hnavi.co.jp/knowledge/blog/pycharm/

・【Pycharm と Python】超簡単に OpenCV の環境を整える話。
https://x.gd/ms167

・PyCharm インストール手順＜Windows 向け＞
https://sukkiri.jp/technologies/ides/pycharm/pycharm-win.html

・3D Ball Tracking in Virtual Environment | OpenCV Python
https://www.youtube.com/watch?v=rPVy-OzWatM

・【Windows10】Python2.7 をインストール
https://algorithm.joho.info/programming/python/27-install-windows10/

・HSV 色空間とは：PEKO STEP
https://www.peko-step.com/html/hsv.html

・Unity：一定時間後に別シーンに遷移（移動）する
https://x.gd/twRsu

・【Unity 入門】ゴールを作ってスコアを数える！ 作りながら覚えるゲーム制作講座 #7（前編）【初心者向け】
https://www.youtube.com/watch?v=c6IGFUoYhB0

・【Unity 入門】文字の表示と変更！ 作りながら覚えるゲーム制作講座 #7（後編）【初心者向け】
https://www.youtube.com/watch?v=bTDSWoqmBno