ポートフォリオ

- GitHub: https://github.com/Furuki0324

目次

※このポートフォリオ内のQRコードはクリックでも動作します。

プレイ動画(YouTube)

「百面夜行」～チームでのゲーム制作～１

ソースコード　プレイ動画

担当箇所:シーン遷移等基本システム、プレイヤー、各種顔パーツ、採点機能、UI　2021/09 ~ 2021/12

- 概要

友達の１人がゲームのアイディアを提案し、それに賛同した６人のチームでゲームを開発しました。

タワーディフェンスがベースとなっており、鼻を破壊しようとする敵を撃破しつつ、福笑いのように好きな位置に目や耳、口を配置し、ゲーム終了時のそれぞれのパーツの位置によってスコアが変わります。

- メンバー構成

プランナー３人、プログラマー（私含め）２人、デザイナー１人

- 参考リンク

プランナーの人たちが作成したPV

参考プレイ動画

↑プレイ中の画面

「百面夜行」～チームでのゲーム制作～２-１

- 制作スタート時の問題点

メンバーが集まって制作を始めることになった際に以下の重大な問題が発覚しました。

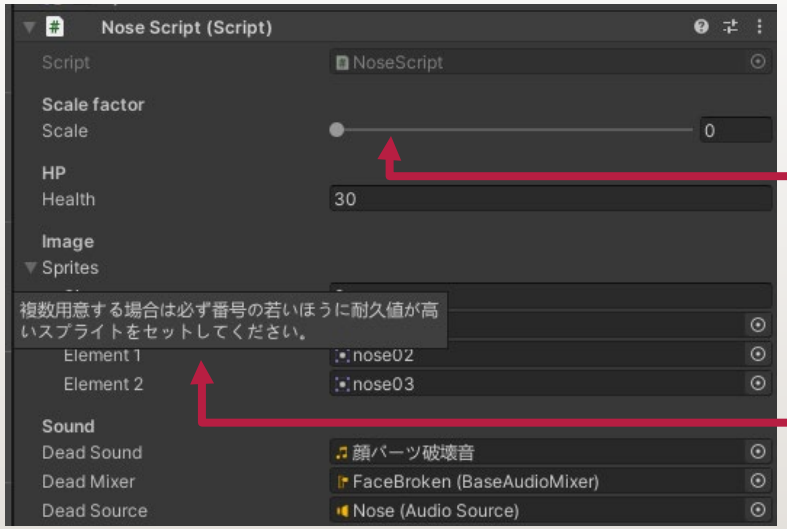
1. ゲームエンジンにUnityを使用する話だったが、Unityの使い方を知っているのは私ともう１人のプログラマーだけだった
2. 共同開発であるためGitHubを使ってプロジェクトデータの共有をしようと提案したが、こちらも使い方を知っていたのは私ともう１人のプログラマーだけだった

この問題を解決するために、ゲームには直接関係しない部分で工夫をしていました。

「百面夜行」～チームでのゲーム制作～２-２

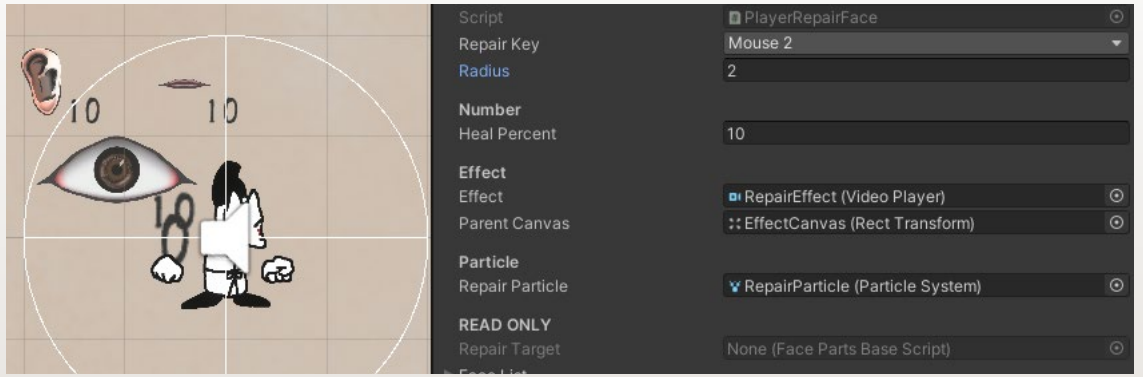
- Unityの使い方に関する問題の解決方法

ゲームの細かな調整はプランナーの担当だったため、Unityの詳しい使い方を知らなくても作業を簡単に出来る環境を目指しました。



入力できる数値の許容範囲があるものはスライダーを使用して入力ミスを防いでいます。

変数の説明を表示することで情報伝達の効率化を図っています。



プレイヤーの周囲のパーツを修理する機能があるため、範囲を可視化することで調整を容易にしています。

この制作活動の以前は個人で制作していたため「動けばそれでよし」と考えている部分がありましたが、チームで活動するためには「わかりやすさが重要」だと気付くきっかけになりました。

「百面夜行」～チームでのゲーム制作～２-３

- GitHubの使い方に関する問題の解決方法

GitHubの使い方は動画の方が伝わりやすいと考えたため、動画を作成してチームに共有しました。

アカウント作成～インストールまで

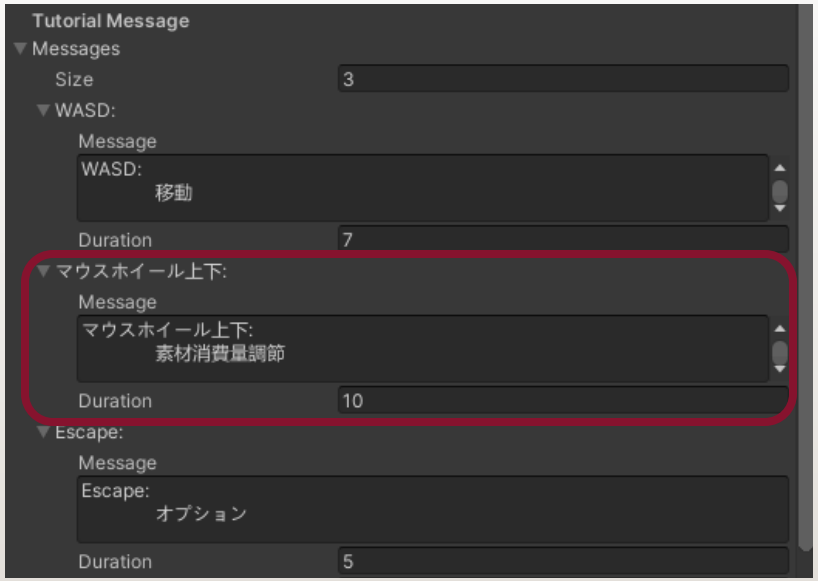
実際にプロジェクトで使用する際の説明

「百面夜行」～チームでのゲーム制作～３-１

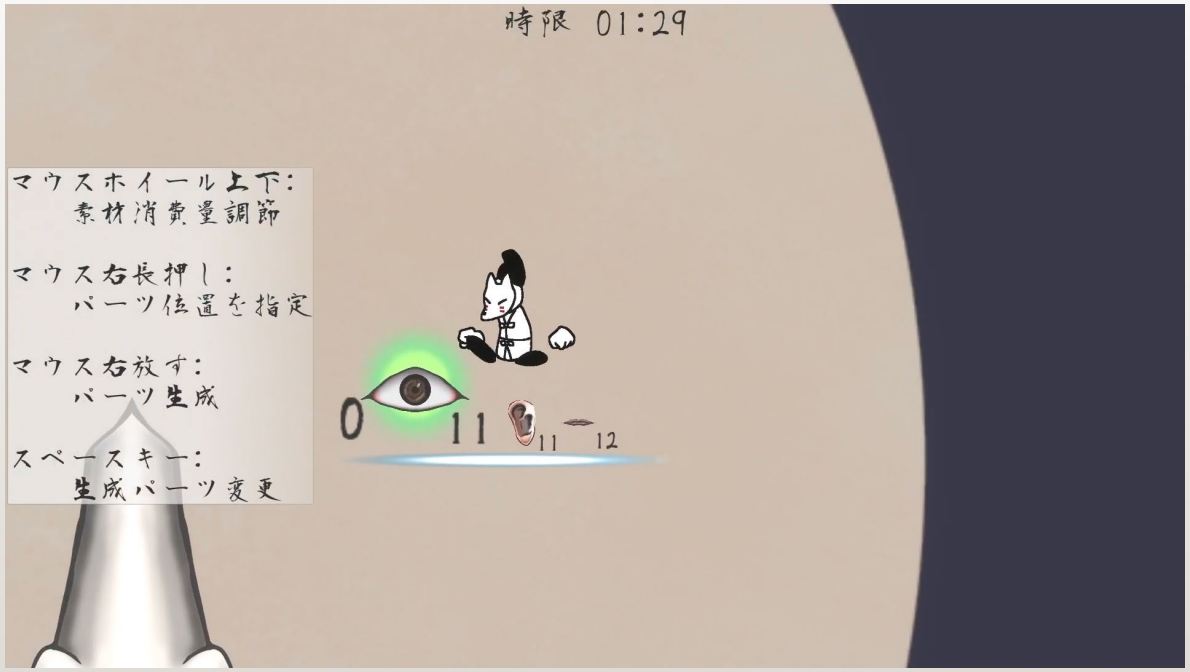
メッセージボックスのソースコード

- プレイヤーが見やすく、開発するチームメイトが使いやすいメッセージボックスの開発

ゲーム開始時の操作説明や実績解除など多目的に使えるメッセージボックスを開発しました。



Messageに入力されたテキストがDuration秒表示されます。



「百面夜行」～チームでのゲーム制作～ 3-2

メッセージボックスのソースコード

- メッセージボックスの工夫点
  1. シングルトンパターンを使用することでメッセージが多重に展開されることを防いでいます。
  2. メッセージの表示にはShowMessage関数を１回呼び出すだけです。
     下記の要素を組み合わせてメッセージの表示を簡略化しています。

```
public struct Message
    {
        public Message(string message, float duration)
        {
            this.message = message;
            this.duration = duration;
        }

        [TextArea]
        public string message;
        public float duration;
    }
```

```
public void ShowMessage(Message message)
    {
        pendingMessageList.Add(message);
        if (!isShowing)
        {
            isShowing = true;
            StartCoroutine(Show());
        }
    }
```

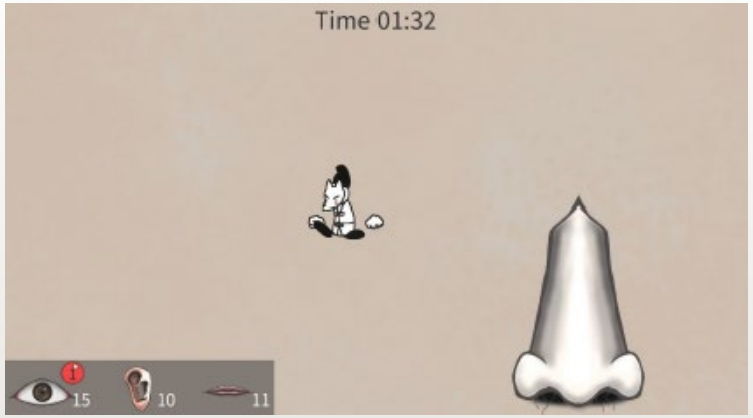
オーバーロード

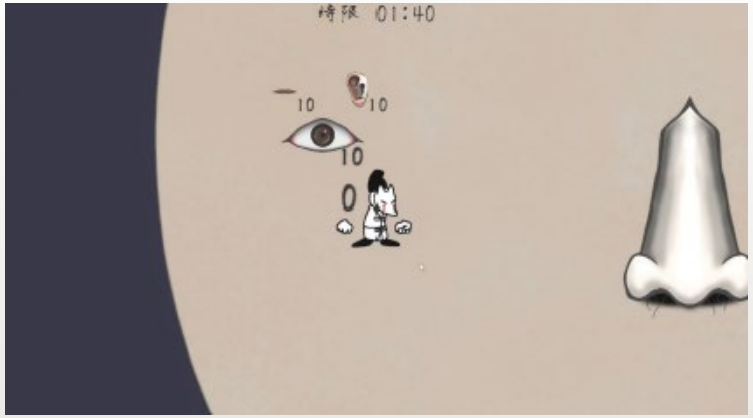
```
public void ShowMessage(string text, float duration)
```

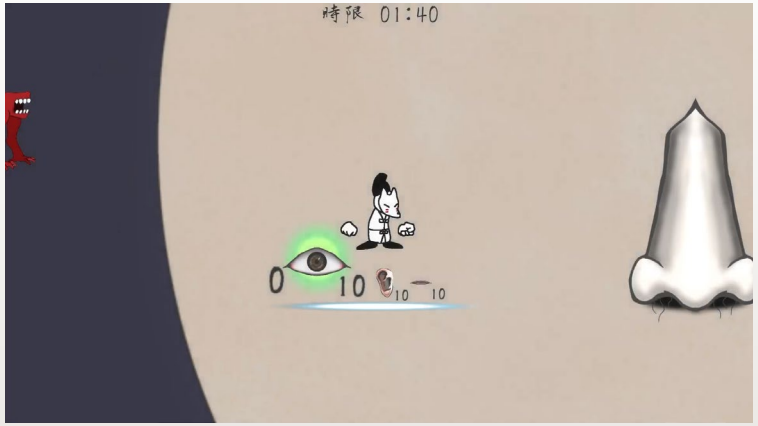
```
private IEnumerator Show()
    {
        Message message;
        while(pendingMessageList.Count > 0)
        {
            message = pendingMessageList[0];
            pendingMessageList.RemoveAt(0);
            // 以降メッセージの表示処理
        }
        isShowing = false;
    }
```

「百面夜行」～チームでのゲーム制作～４

- UIデザインの変更　　よりコンパクトで使い勝手の良いUIを目指しました。







- 初期型
- 画面左下に固定。
- イラストをクリックしてパーツを選択
- 素材の消費量は同じイラストをクリックするたびに１増加。消費量は赤い円の中に表示。

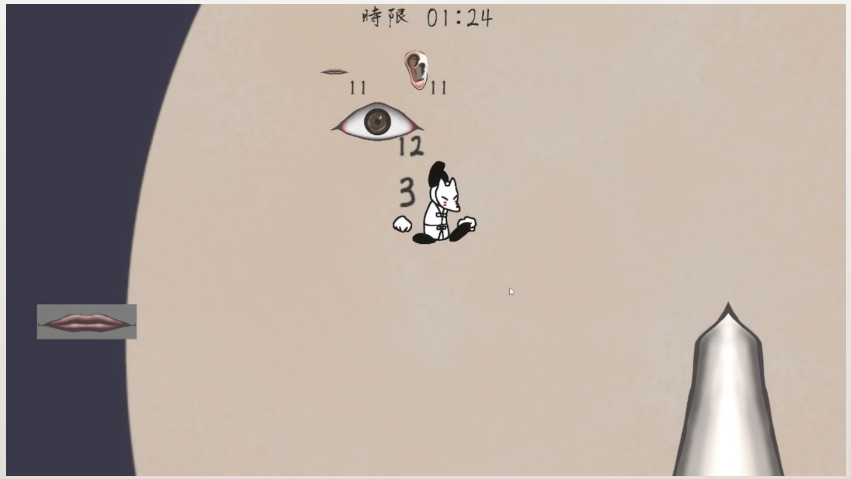
- 中期型
- プレイヤーに追従して動く。
- スペースキーでパーツ切り替え。三角形の下側が選択中のパーツで、時計回りに切り替わる。
- 素材の消費量はマウスホイールの上下で増減。

- 後期型
- 中期型のレイアウトを変更。
- 新たに強攻撃が実装されたため、そのクールタイムを示すバーを追加。クールタイム中は色が暗くなる。またクールタイム終了時に効果音を再生する。

「百面夜行」～チームでのゲーム制作～５-１

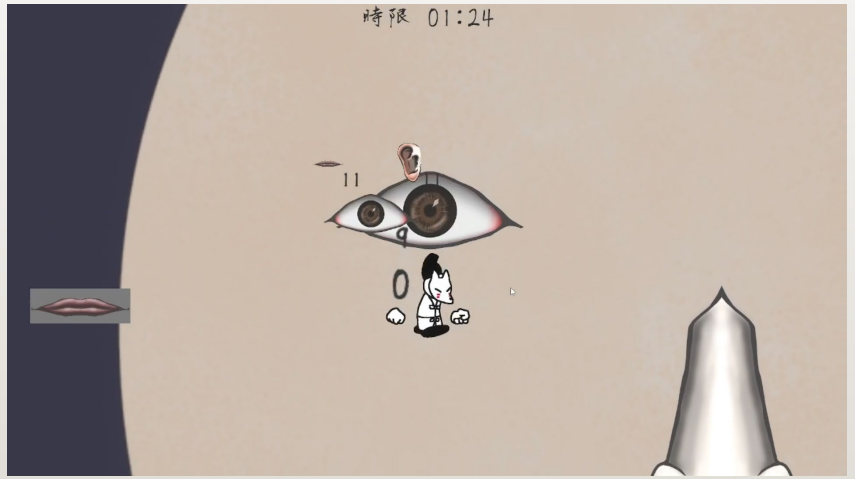
- 顔パーツ生成時の工夫

開発の初期段階では顔パーツとプレイヤーは衝突しない設定でしたが、プランナーから衝突するようにしてほしいと要望があったため、変更したところ下記の問題が発生しました。



生成前

顔パーツはプレイヤーの位置に生成されるため、プレイヤーがはじき出されてしまう。

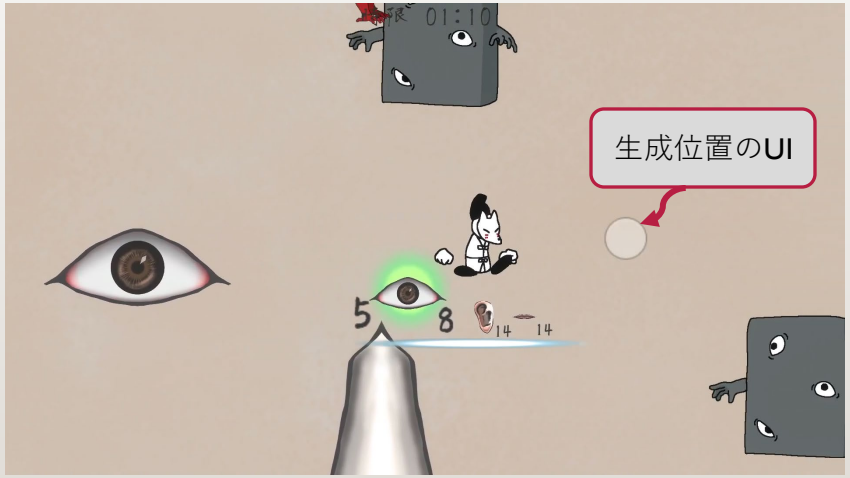


生成後

「百面夜行」～チームでのゲーム制作～5-2

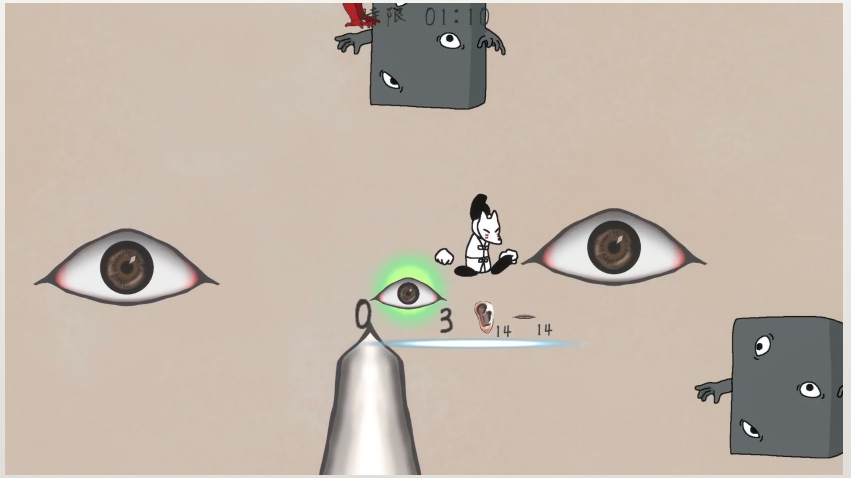
- 顔パーツ生成時の工夫

前ページの問題の解決策として、顔パーツの生成位置をプレイヤーの進行方向の少し離れた位置に変更しました。同時に生成位置を示すUIを実装し、プレイヤーが簡単に位置を調節できるようにしました。



生成前

生成ボタンを押している間UIが表示され、ボタンを放すとUIが非表示になり顔パーツが生成されます。



生成後

ボディトラッキングで操作するゲーム１

ソースコード

プレイ動画

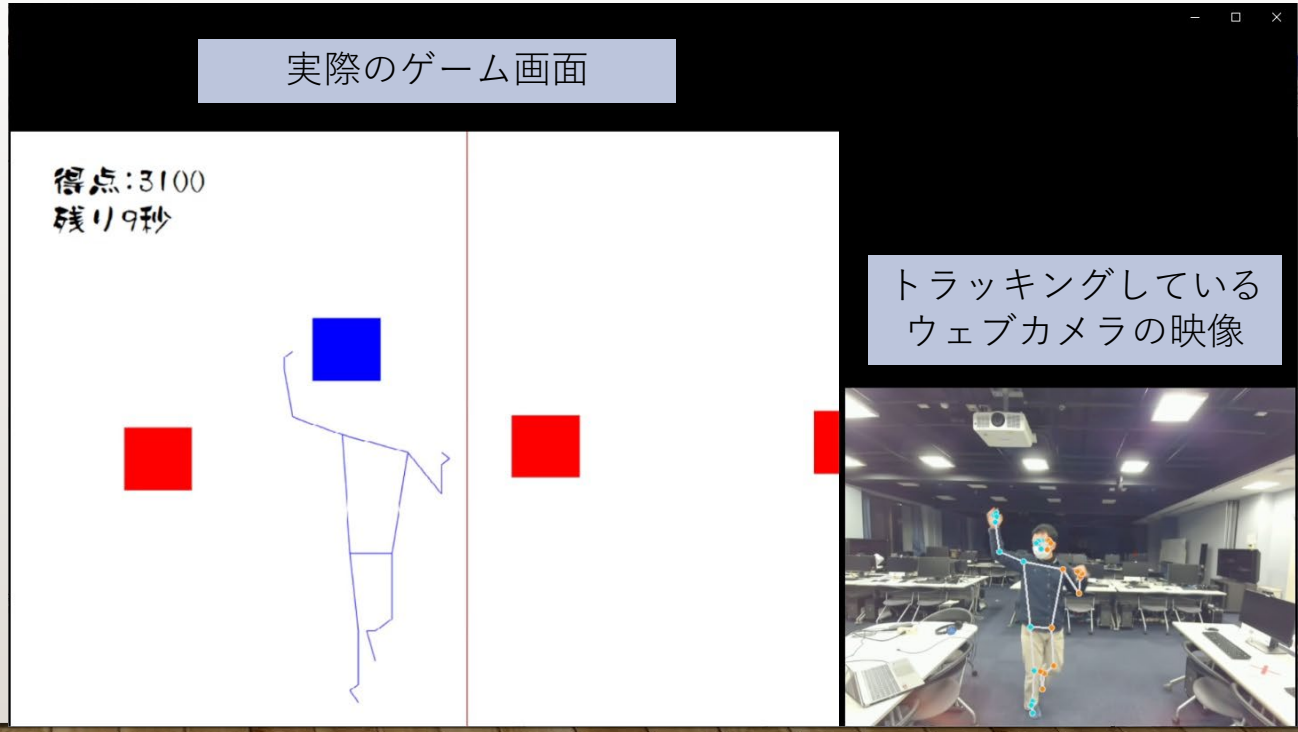
2022/01 ~ 2022/02

- 概要・制作意図

YouTubeでMediaPipeというフレームワークの解説動画を見たことをきっかけに、この技術を使用したゲームを作ることにしました。
またゲームエンジンを使わないゲーム開発にも興味があったので、そちらにも初めて挑戦した作品です。

- 使用した技術

Direct2D
MediaPipe　https://google.github.io/mediapipe/
UDP通信(Pythonでトラッキングを実行し、取得したデータを一度文字列に変換してC++のゲームプログラムに送っています。)



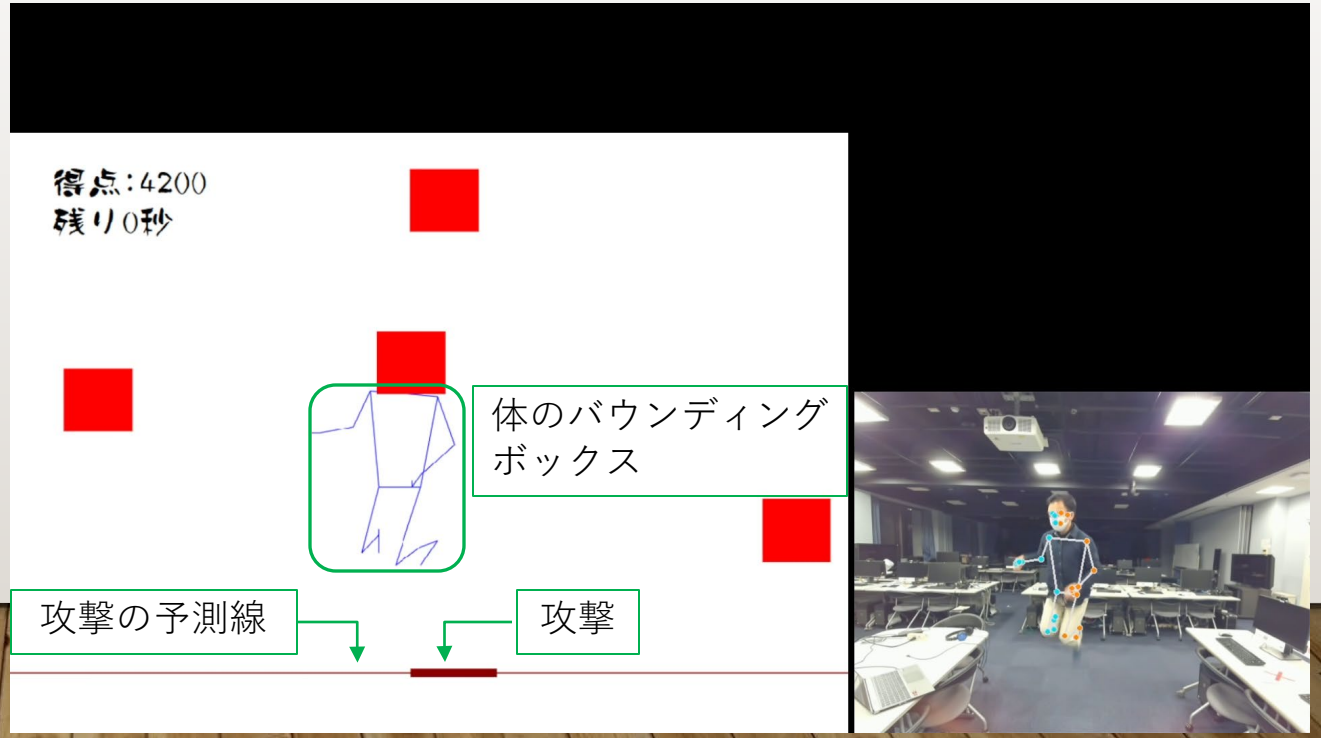
ボディトラッキングで操作するゲーム２

- 全身を使ったダイナミックなプレイ体験

この作品で初めてボディトラッキングをゲームに組み込んだため、指先の繊細な操作よりもジャンプや体のひねりのようなダイナミックでわかりやすい動きを設計に組み込みました。

- 攻撃と体の接触判定

体の接触判定にはバウンディングボックスの考え方を使用しています。体のバウンディングボックスと攻撃の長方形が重なった場合に被弾判定を発生させています。



RPG開発に使用するツール開発１

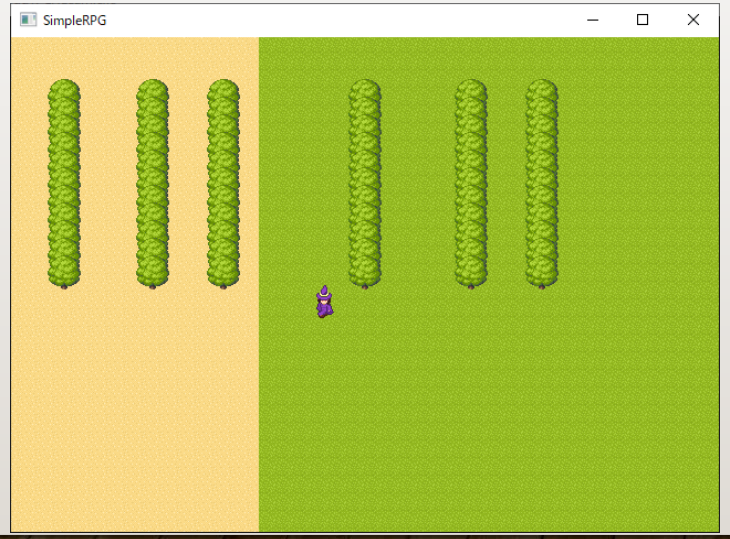
ソースコード

2022/04 ~2022/06

- 概要

プログラミングの勉強の為にC++で開発していた2Dゲームの支援ツールです。
マップをCSVファイルで管理していましたが、CSVファイルの番号指定で時間がかかっていたため時間短縮のためのツールを作ることにしました。

→
ゲームを実行できる段階まで開発した際の画面です。



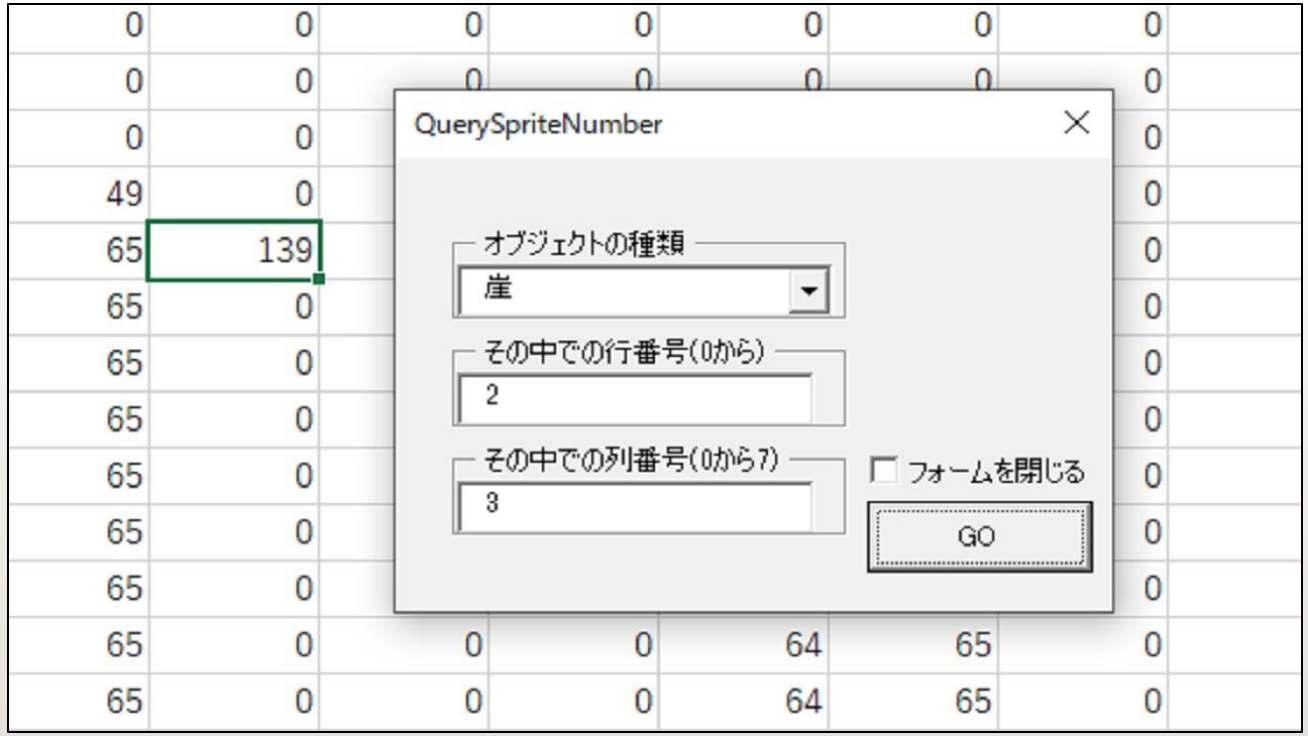
→
使用したマップチップのセットの一部です。
16x16ピクセルのチップがまとめられて128x4672ピクセルになっています。

RPG開発に使用するツール開発２-１

- 初期段階

VBAを使ってエクセルにマクロを作成しました。

必要な項目を設定して実行（GO）するとアクティブセルに番号が入力されます。



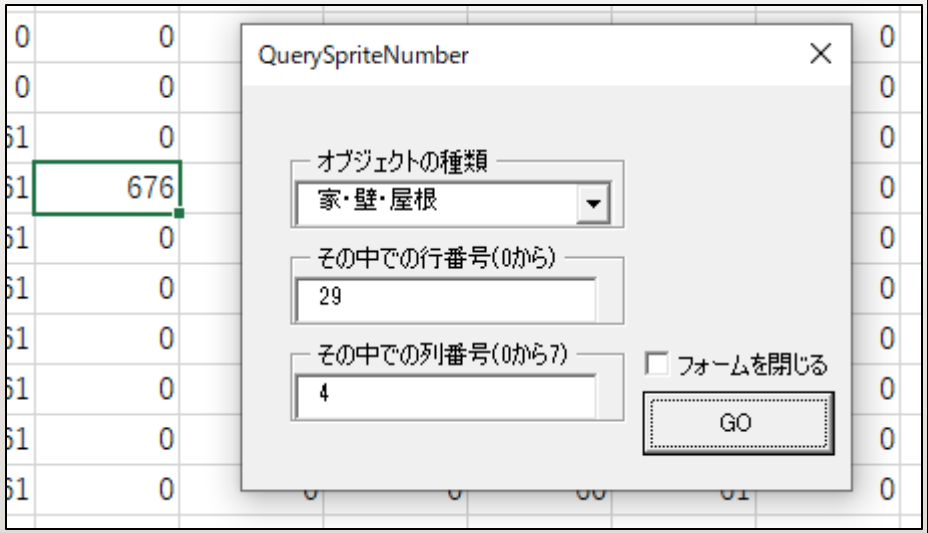
RPG開発に使用するツール開発２-２

- 問題点

直接計算して入力するよりは簡単に入力できるものの、画像２のように

「家・壁・屋根」グループ内に多くのチップが含まれているときに行番号が大きくなってしまい、時間がかかったり数え間違いの原因になったりしてしまう。

↓画像１



→画像２

RPG開発に使用するツール開発３

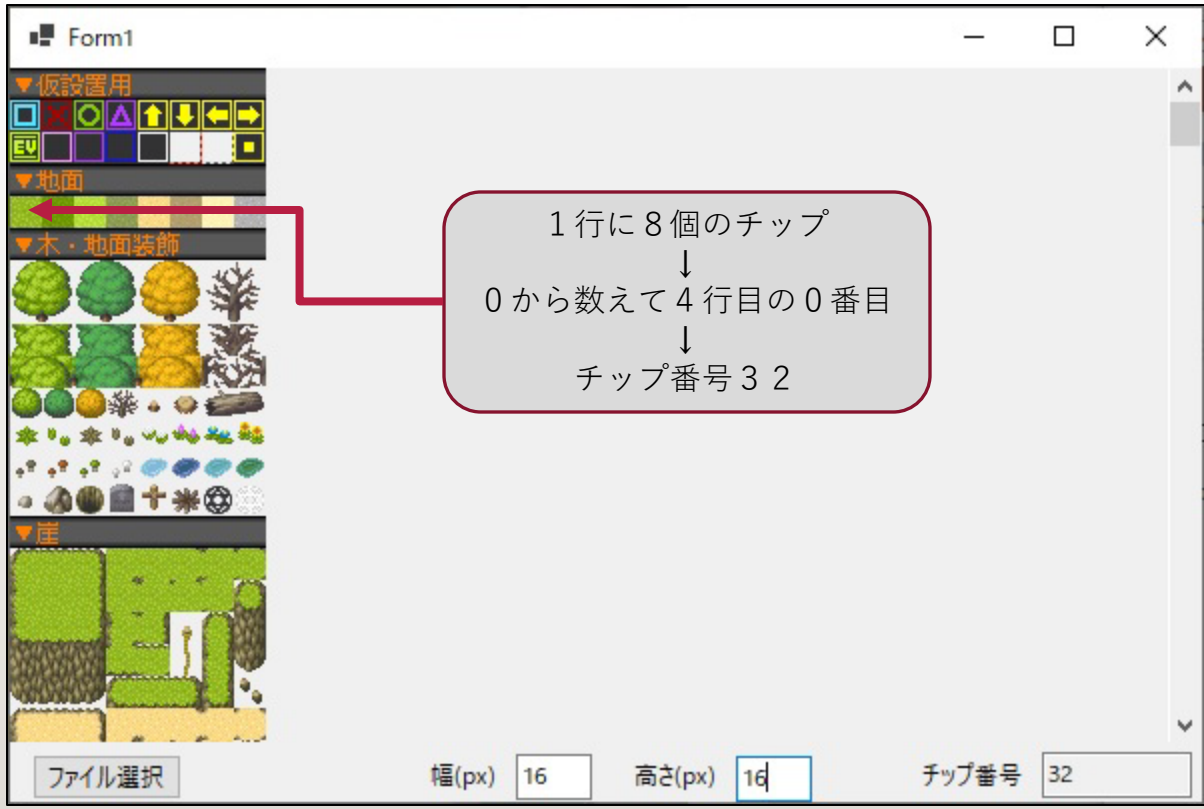
この段階のソースコード

- VBAのマクロから改善

先ほどの問題を解決するために、C#を利用してアプリケーションを作成しました。

下部にチップのサイズを入力してから画像の使用したい部分をクリックすると、

クリックした部分の番号が右下に表示されます。



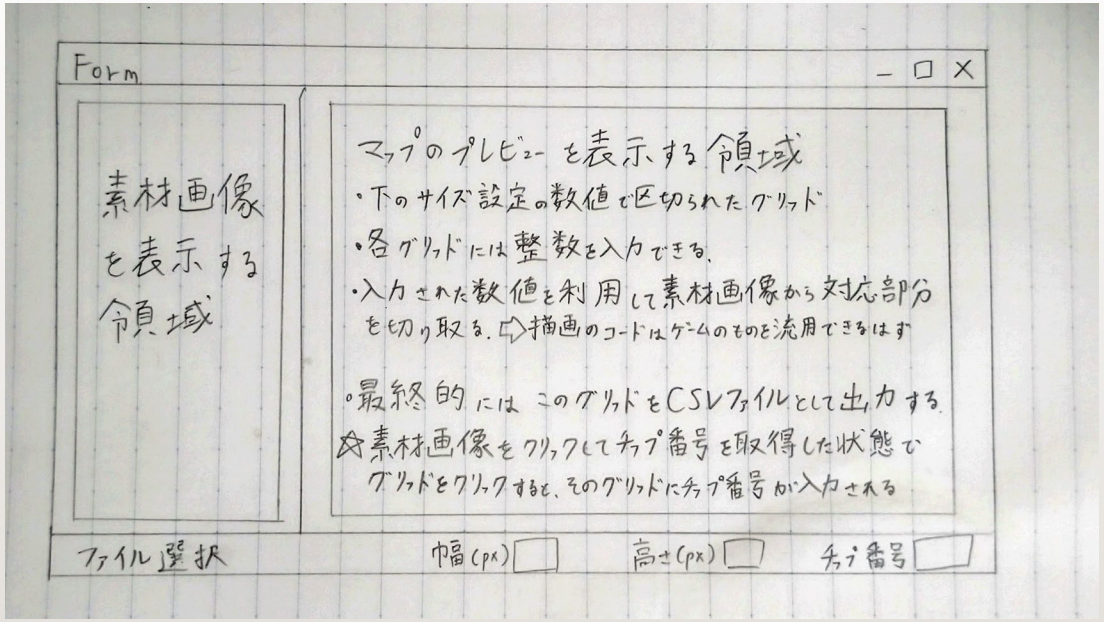
RPG開発に使用するツール開発４-１

- 更なる改善案

C#アプリにしたことで数え間違いはなくなりましたが、エクセルを開いて数値を入力する手間が増えてしまいました。

また今までは編集したマップを確認するためにゲームを起動する必要もあったため、チェックに時間がかかっていました。

これらを解決するために先ほどのC#アプリにCSVファイルの出力機能やプレビュー機能を持たせたツールを考え、開発しました。



RPG開発に使用するツール開発４-２

この段階のソースコード

解説動画

- マップのプレビューを表示している裏側にDataGridViewが隠れています。プレビューはDataGridViewの値を読み取って表示する色を選択しています。
- DataGridViewの編集したい部分をクリックするだけで値が切り替わります。
- DataGridViewの値が変わるとプレビューの描画が始まります。負担軽減のため、再描画は値が変わった部分だけ実行されます。

- 「新規」「保存」「読み込み」「出力」の４コマンドが含まれています。
- 「保存」では素材画像のパスやDataGridViewを変換したCSVファイルのパス等をJSON形式で記録します。
- 「読み込み」では保存したJSONファイルを読み込み、以前の状態を復元します。