ポートフォリオ

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

目次

「百面夜行」～チームでのゲーム制作～

GitHub: https://github.com/Furuki0324/Hyakumenyakou_ZemiA　　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。

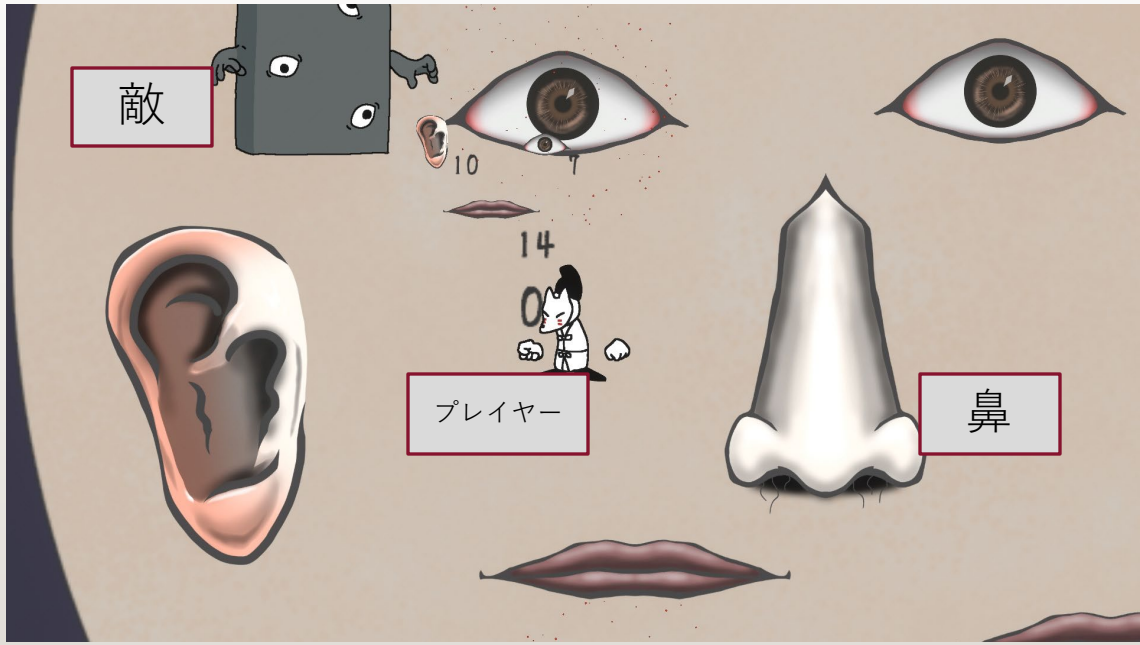
- メンバー構成

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～

- 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。

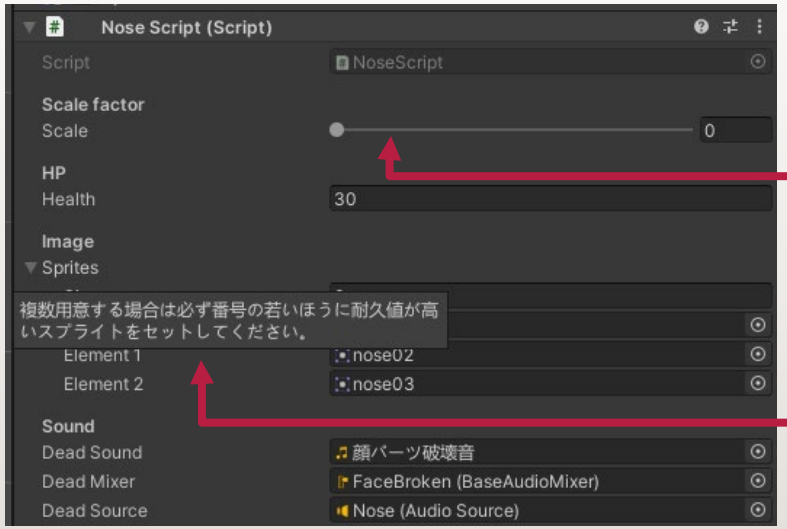
1. ゲームエンジンにUnityを使用する話だったが、Unityの使い方を知っているのは私ともう１人のプログラマーだけだった
2. 共同開発であるためGitHubを使ってプロジェクトデータの共有をしようと提案したが、こちらも使い方を知っていたのは私ともう１人のプログラマーだけだった

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～

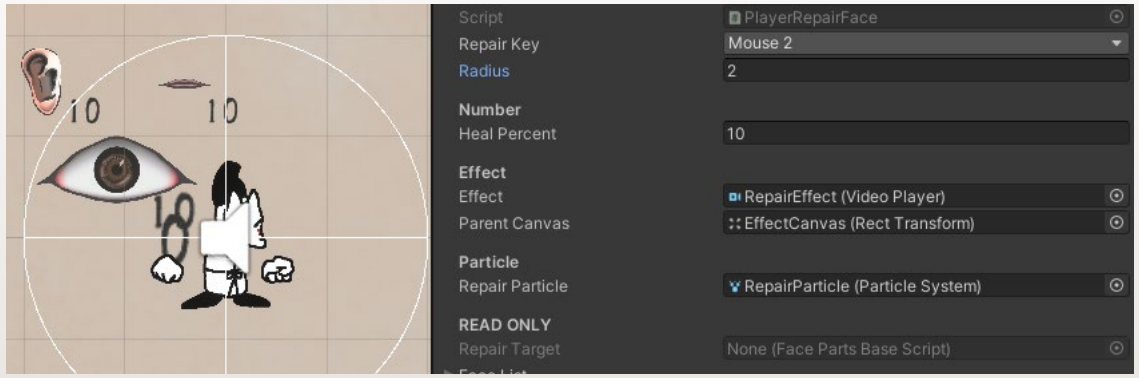
- **Unity**の使い方に関する問題の解決方法

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何に注意するべきか考えるきっかけになりました。

「百面夜行」～チームでのゲーム制作～

- 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

Ctrlキーを押しながら画像クリックでYouTubeに移動します。

実際にプロジェクトで使用する際の説明

ボディトラッキングで操作するゲーム

GitHub(ゲーム部分): https://github.com/Furuki0324/SimpleUDP　　GitHub(MediaPipe): https://github.com/Furuki0324/MediaPipe_py

2022/01 ~ 2022/02

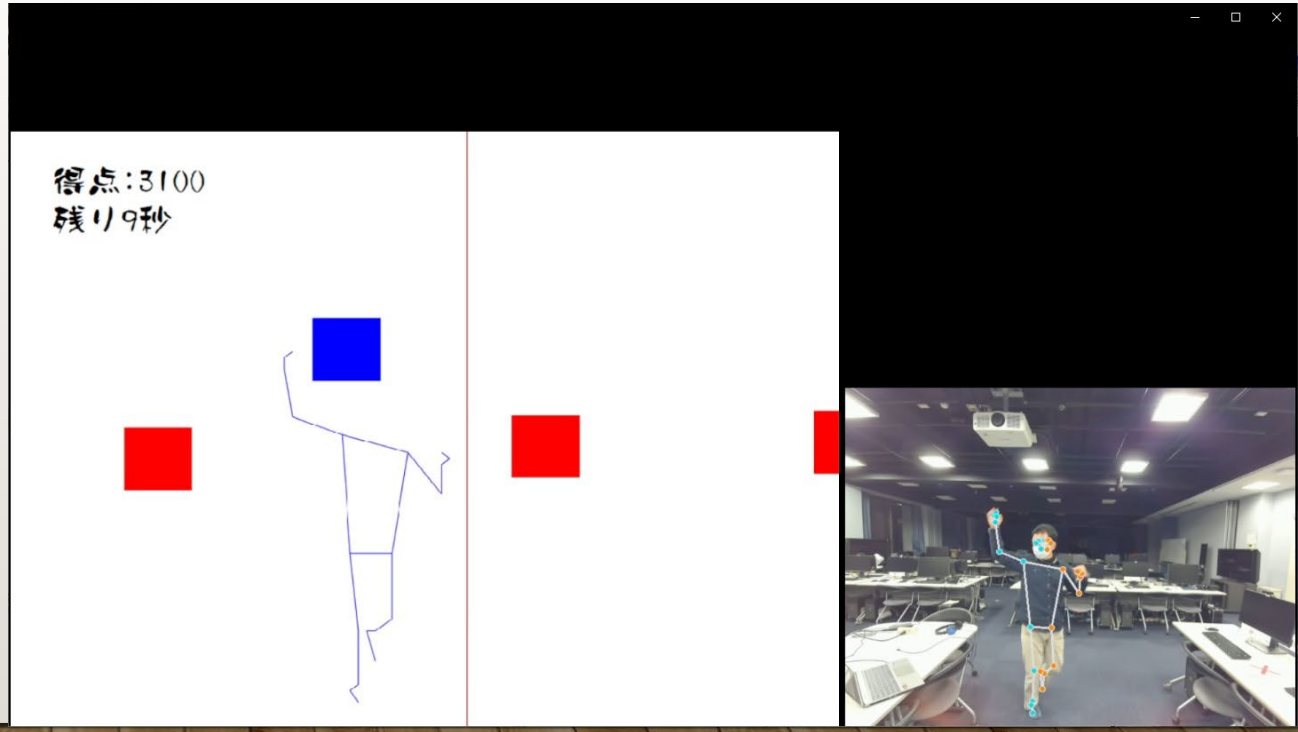
- 概要

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、そちらにも挑戦した作品です。

- 使用した技術

Direct2D
MediaPipe　https://google.github.io/mediapipe/
ソケット通信(UDP)

https://youtu.be/JICmLMhPYVY



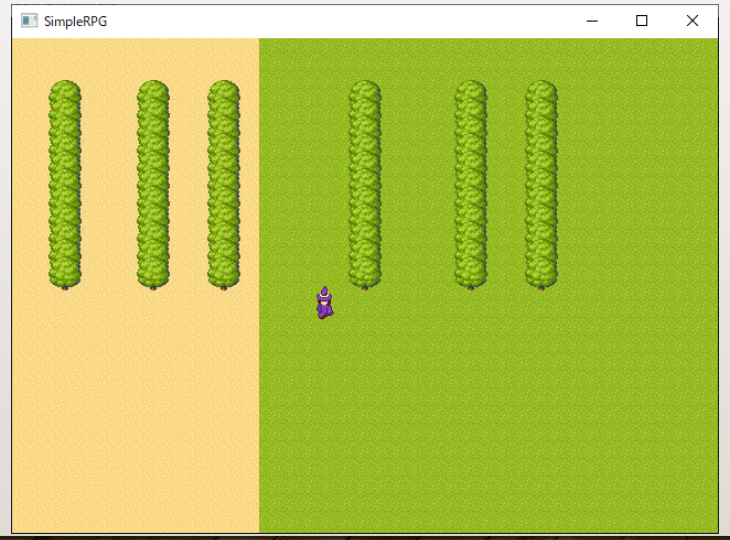
RPG開発に使用するツール開発１

GitHub(作成中のゲーム): https://github.com/Furuki0324/PCHLearn　GitHub(ツール): https://github.com/Furuki0324/MapMaker

2022/04 ~

- 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。



【左の画像】
ゲームを実行できる段階まで開発した際の画面です。

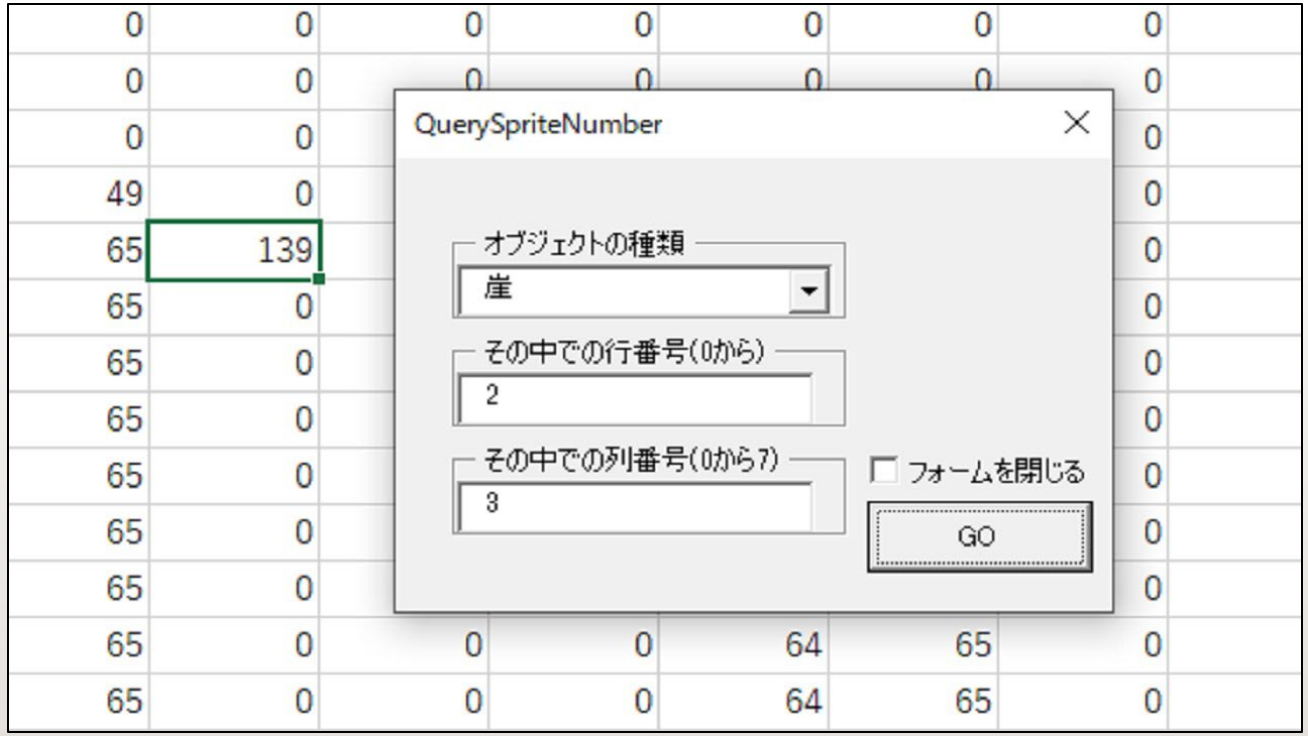
【右の画像】
使用したマップチップのセットの一部です。
16x16ピクセルのチップがまとめられて
128x4672ピクセルになっています。

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。



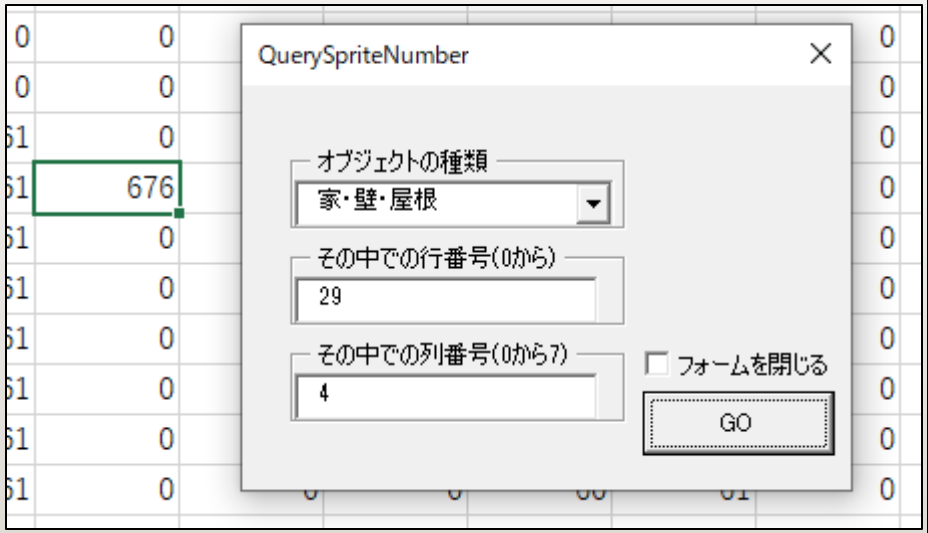
RPG開発に使用するツール開発２-２

- 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

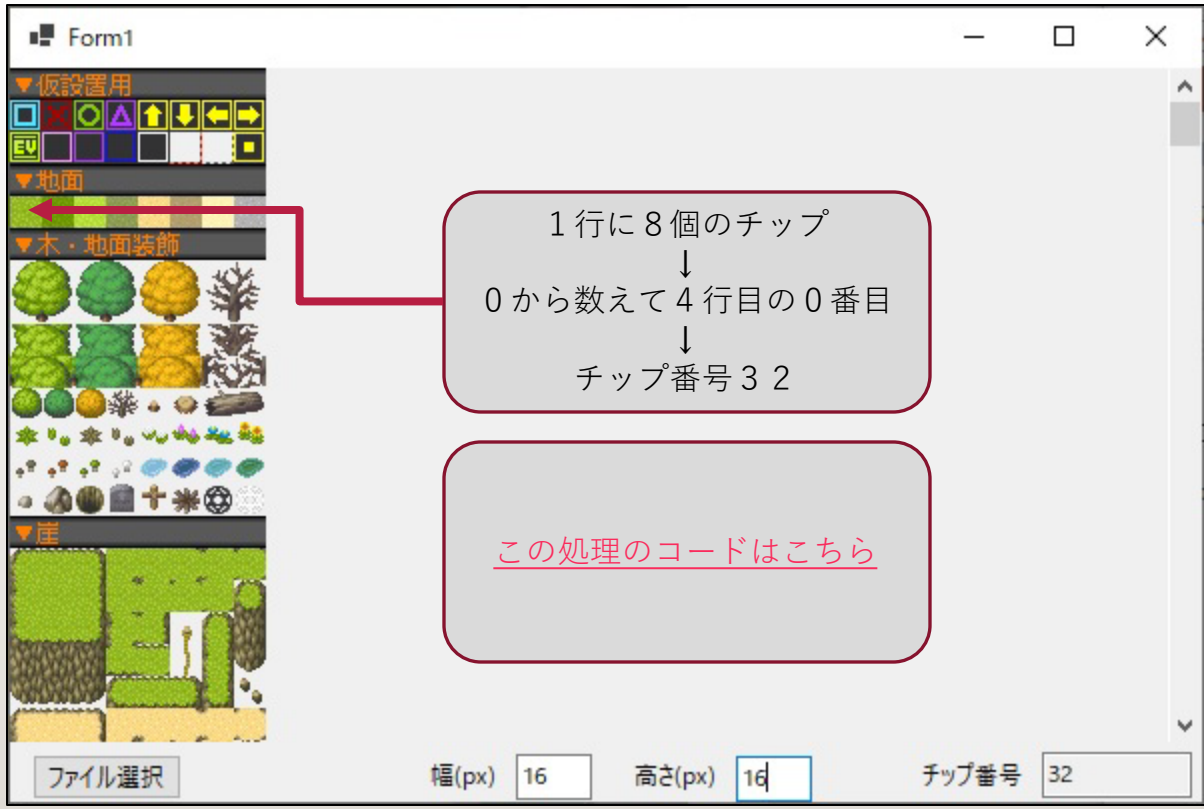
この段階のソースコード: https://github.com/Furuki0324/AppForMapchip

- 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。



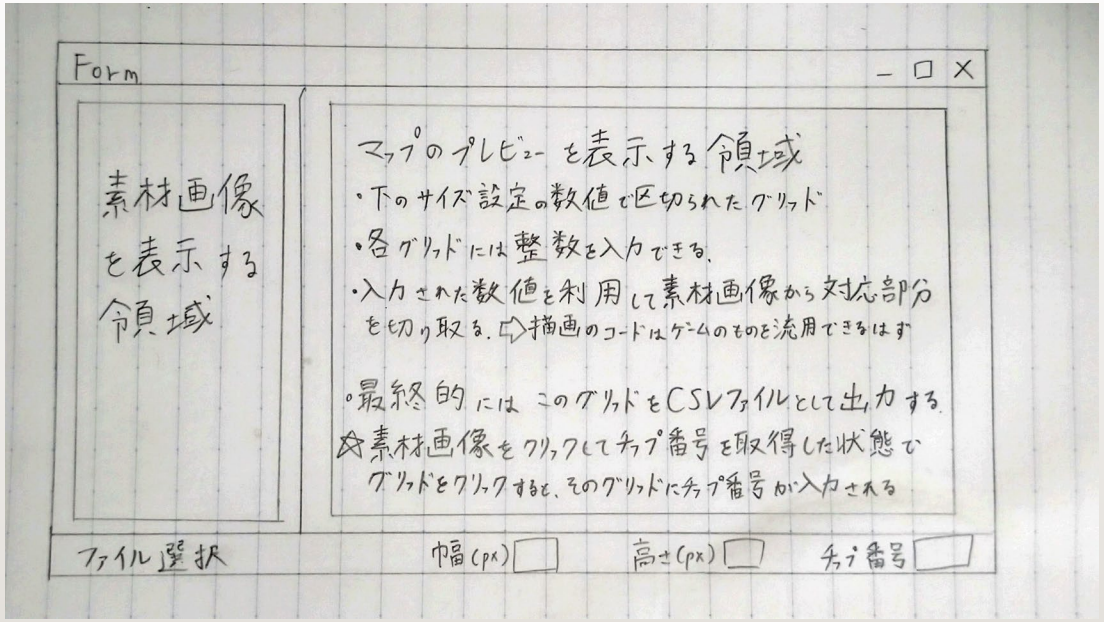
RPG開発に使用するツール開発４-１

- 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ったが、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、開発しています。

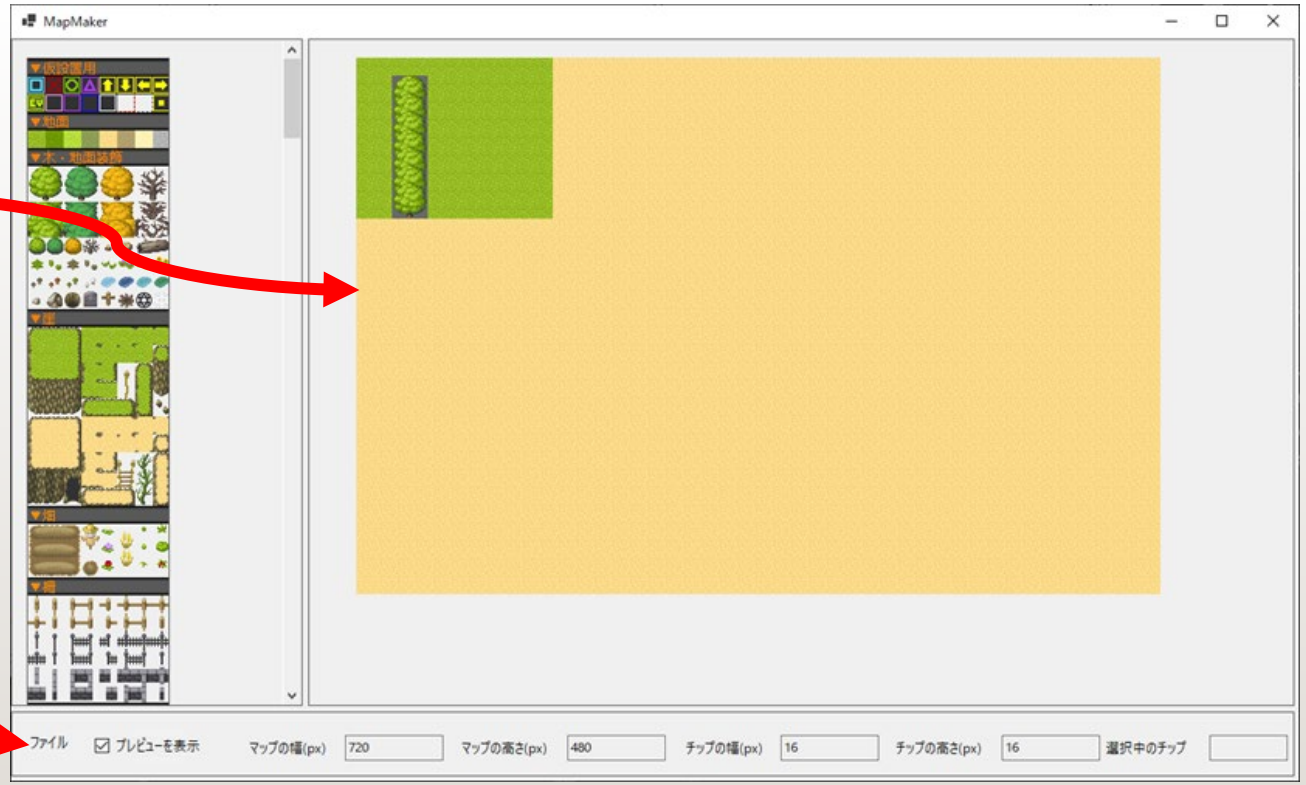


RPG開発に使用するツール開発４-２

この段階のソースコード https://github.com/Furuki0324/MapMaker

- 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。



画像をクリックするとYouTubeの解説動画へ繋がります。

１１ページのコード

```
private void OnImageClick(object sender, EventArgs e)
        {
            //マウスと画像の原点のスクリーン上の座標を取得
            Point mousePoint = MousePosition;
            Point pictureOrigin = PointToScreen(pictureBox1.Location);

            //取得した座標からマウスが画像上のどの位置にいるのか求める
            int x = mousePoint.X - pictureOrigin.X;
            int y = mousePoint.Y - pictureOrigin.Y;

            int chipWidth;
            int chipHeight;
            int spritePerRow;

            //テキストボックスに後の計算が正常に実行されない値が入力されていた場合は中断する
            if(WidthPixel.TextLength == 0
                || WidthPixel.Text == "0"
                || !int.TryParse(WidthPixel.Text, out chipWidth)
                || HeightPixel.TextLength == 0
                || HeightPixel.Text == "0"
                || !int.TryParse(HeightPixel.Text, out chipHeight))
            {
                SpriteNumber.Text = "NaN";
                return;
            }

            //画像の１行の中に何枚のチップがあるか求める
            spritePerRow = pictureBox1.Width / chipWidth;

            //クリックしたチップの番号を求める
            int spriteNum = (y / chipHeight) * spritePerRow + x / chipWidth;
            SpriteNumber.Text = spriteNum.ToString();
        }
```