

自己紹介



いちの りょうま

名前：一野 諒馬

生年月日：2003年3月17日

学校：名古屋工学院専門学校

出身地：岐阜県岐阜市

自分の好きを詰め込んだゲームを、沢山の人に遊んで貰い、人から評価された時の喜びからゲーム制作を好きになりました。

趣味・特技

■プラモデル制作

作ることの楽しさを知ったきっかけがプラモデル出した。簡単に作れて、達成感を得られるので楽しいです。他には、墨入れやデカール等でデコレーションが出来て、自分の満足のいくところまでこだわられる点や、他の人の作品を鑑賞して楽しめる点も好きな理由です。

使用できるソフト・言語

■ Unity / C

プレイヤーや敵の挙動も作りますが、得意なものはUIの処理です。UniRxを使い参照関係をきれいにしたり、DoTweenでUIを動かします。

■ Git / SourceTree

主にSourceTreeを利用してプロジェクトの共有をしています。

■ ADX LE

主にUnityでサウンドの管理ツール兼サウンドプレイヤーとして利用しています。

■ SpriteStudio

過去の制作で2Dアニメーション作成の為に利用しました。

■ Aseprite

ドット絵専用のイラストツールです。個人制作等でイラストを自作しています。

WarlockHero



日本ゲーム大賞アマチュア部門に向けて制作、その後ゲームクリエイター甲子園に向けて制作を行いました。

プレイヤーは魔法使いとなり、毎回形が変わるダンジョンに入り、暴れているモンスター達を様々な魔法を駆使して倒します。そして、最奥にいるボスを倒すことがゲームの最終目標です。

ジャンル：ローグライクシューティング

制作環境：Unity 2022.3.5f1

VisualStudio 2019

GitHub / SourceTree

制作期間：2023年9月6日 ～ 2024年1月24日

制作人数：5人（プログラマー 5人）

担当箇所：敵

（雑魚敵 3体 / 中ボス 1体）

テキスト表示

（チュートリアルステージ、NPCの頭上に出る吹き出し）

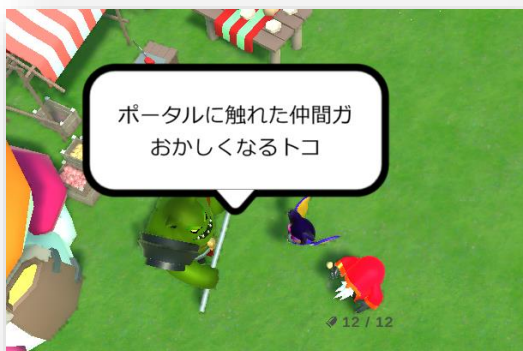
作品ドライブ



WarlockHero ~ 工夫した点 ~

■テキスト表示

ゲーム内で表示されるテキストの処理を作成しました。



・吹き出し

枠がテキストに合わせて
サイズが変わるようにしました。



・会話パート

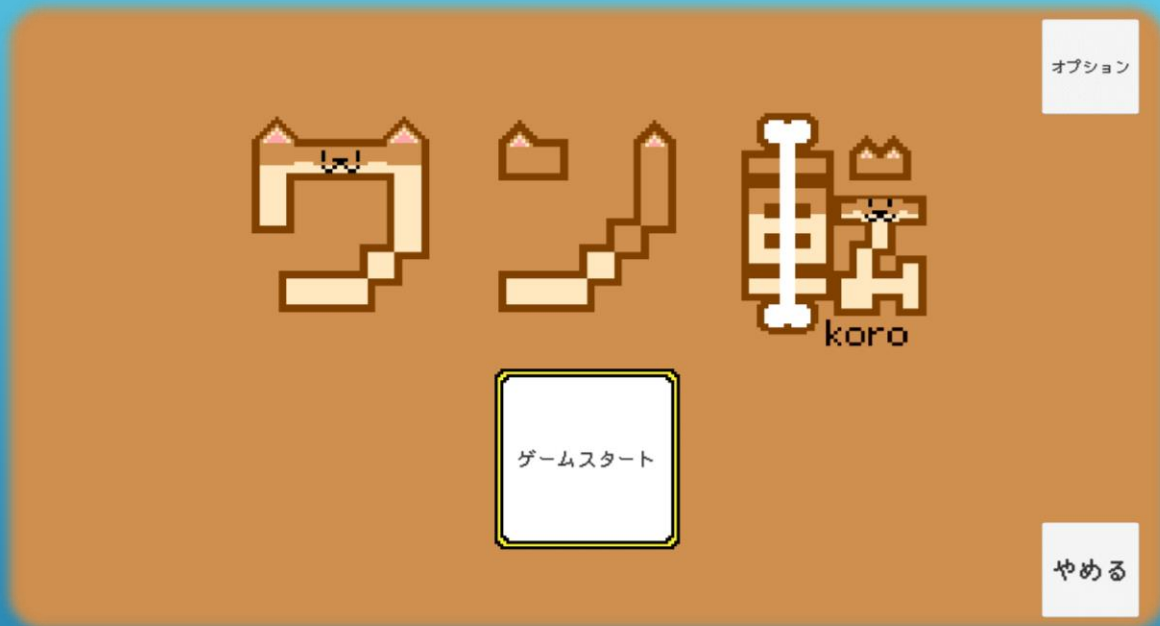
一番上に、話している人の名前。
その下に、会話内容を表示するようにしました。
テキストのバックに置くImageは半透明にして、
後ろが少し見えることで、主張しすぎない見た目になりました。

WarlockHero ~ 工夫した点 ~

- エディター
 - テキスト内容を、ScriptableObjectで編集できるようにした。
 - ReorderableListを使い、再生したいテキストの順番を入れ替えられるように拡張した。
 - テキスト再生速度、次のテキストへの切り替え時間を区切りごとに調整できるようにした。



ワン転 (わんころ)



個人的にスマホゲームを作りたいと考えていた時に、過去の作品のリファクタリングの話を聞いて、専門1年生の時に制作したゲームをもとに作成したゲームです。

作るきっかけは授業でしたが、授業外でも制作を進めていきました。

Androidでリリースしています。

ジャンル：ランゲーム

制作環境：Unity 2022.3.25f1
VisualStudio 2022
GitHub / SourceTree
ADX LE
Aseprite

制作期間：2024年5月2日 ～ 2024年6月31日

制作人数：2人（プログラマー2人）

役割：リーダー

担当箇所：企画、プログラム全般、イラスト全般、サウンド管理



ワン転（わんころ） ～ 工夫した点～

■オブジェクトプール

1年生の時に作った作品では、オブジェクトを生成すると画面が一瞬固まりました。
それを修正するために、今回は自作でオブジェクトプールを作成しました。

自作のオブジェクトプールは、設定したキャパシティを超えるまではInstantiateをしますが、キャパシティを超えると古いオブジェクトから再利用する処理になっています。

最初、作成したときはListを使用していましたが、「古いものから再利用する」という処理から、Queueのほうが処理が軽いとわかり、Queueで作り直しました。

```
public class ObjectPool
{
    // オブジェクトのキュー
    private Queue<GameObject> queue = new Queue<GameObject>();
    private Queue<PoolBase> poolBaseList = new Queue<PoolBase>();
    // プレハブ
    private GameObject prefab;
    // キャパシティ
    private int capacity;

    9 個の参照
    public ObjectPool(GameObject obj, int Capacity)
    {
        prefab = obj;
        capacity = Capacity;
    }

    /// <summary>
    /// 生成
    /// </summary>
    /// <param name="position">座標</param>
    9 個の参照
    public void Instance(Vector3 position)
    {
        GameObject obj;
        PoolBase poolBase;
        // キャパシティを超えていないか確認
        if (queue.Count < capacity)
        {
            // インスタンス生成
            obj = Object.Instantiate(prefab, position, Quaternion.identity);
            poolBase = obj.GetComponent<PoolBase>();
            poolBase.ObjPool = this;
        }
        else
        {
            // キューの先頭のオブジェクトを再利用
            obj = queue.Dequeue();
            obj.transform.position = position;
            obj.SetActive(true);
            poolBase = poolBaseList.Dequeue();
            poolBase.OnGenerate();
        }
        queue.Enqueue(obj);
        poolBaseList.Enqueue(poolBase);
    }
}
```

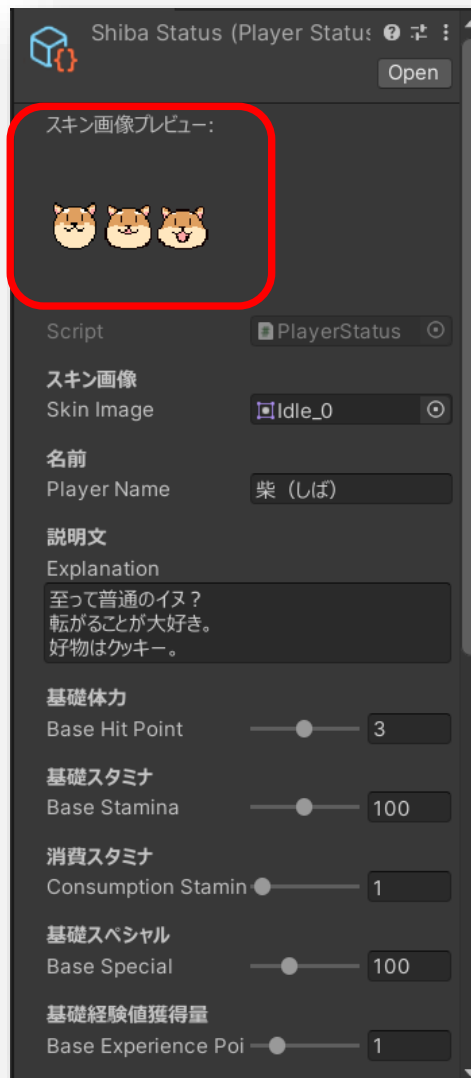
ワン転（わんころ）

～工夫した点～

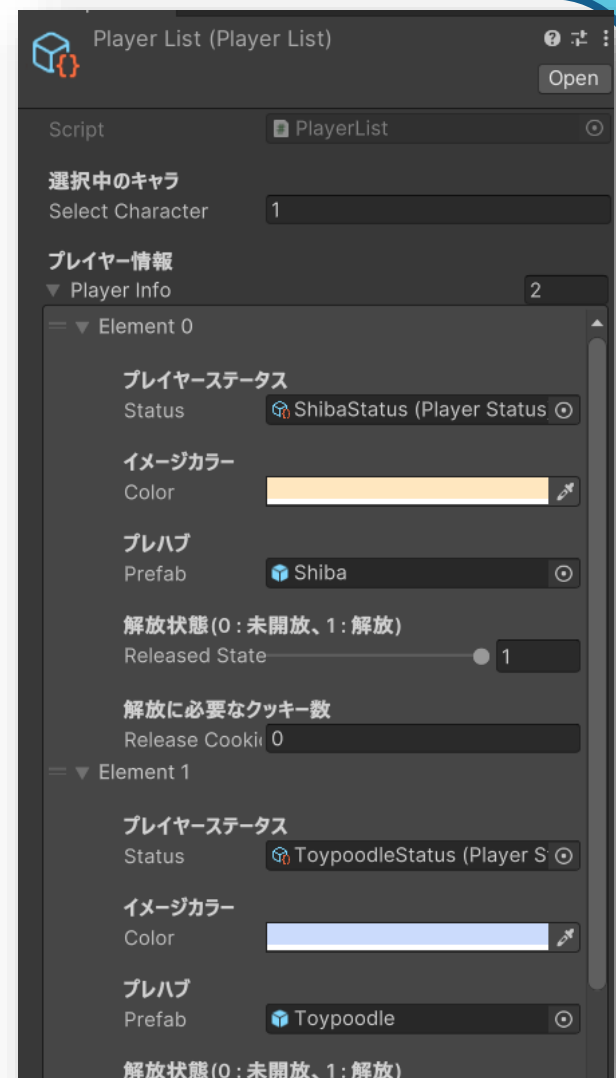
■ キャラクターの追加

キャラクターごとのステータスをScriptableObjectで管理し、数値の調整をやすくしました。

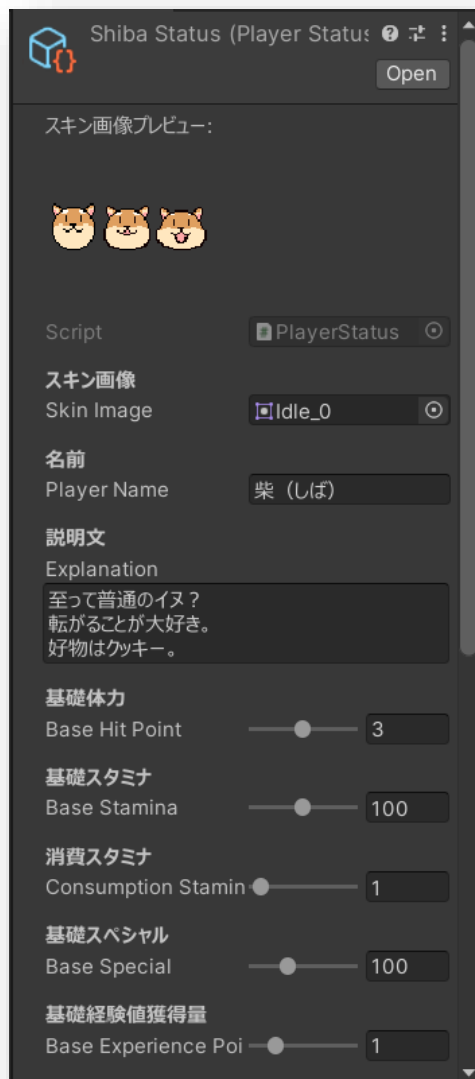
一番上には、そのキャラクターを設定したときに表示する画像をプレビューとして表示されるようにエディター拡張をしました。



キャラクターのデータをScriptableObjectで管理し、ここに必要なデータを追加するだけで、キャラクターを簡単に追加できるようにしました。

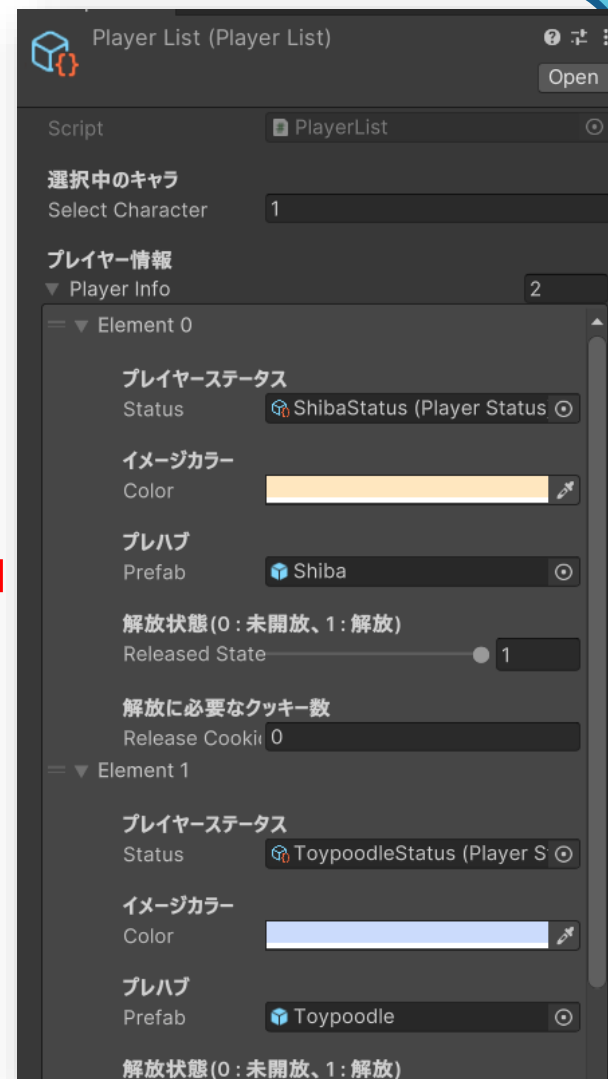


ワン転（わんころ） ～ 工夫した点～



設定したステータスデータは
左側に表示される

データをリストに追加したら
自動的に右のスクロールビュー
に追加される



ワン転（わんころ） ～ 工夫した点～

■ キャラクターの追加

キャラクターの処理も「PlayerAction」を継承して、各操作に対する関数をオーバーライドすることで簡単に作成できるようにしました。

プレイヤーの出来る操作は、ジャンプ、急降下、スライディング、スペシャルの4つです。

その中で、スライディングの処理はボタンを押したときと、離れたときの2つの処理を用意しました。

理由は2つあり、1つ目は押している間にスライディング用の当たり判定を出し、離れたときに判定を消すという処理にしているからです。

2つ目はUIのボタンを使用しているからです。ボタンの長押し処理は、押したときと離れたときでしか処理ができないからです。

```
public class PlayerDefaultAction : PlayerAction
{
    3 個の参照
    public override void GroundUp()
    {
        base.GroundUp();
        Vector3 setVelocity = new Vector3(rb.velocity.x, 0f, rb.velocity.z);
        SEPlayer.Instance.PlaySound(SEType.Player, 0);
        // 接地状態でジャンプ
        rb.AddForce(Vector3.up * jumpPower, ForceMode.Impulse);
        ChangeState(PlayerState.Jump);
    }

    3 個の参照
    public override void AirUp()
    {
        base.AirUp();
        // 空中で急降下
        rb.AddForce(fallSpeed * Vector3.down, ForceMode.Impulse);
        ChangeState(PlayerState.Swoop);
    }

    3 個の参照
    public override void GroundOnDown()
    {
        base.GroundOnDown();
        // 接地状態でしゃがみ
        // ボタンが押されているときの処理
        spriteTransform.eulerAngles = new Vector3(0f, 0f, 0f);
        ChangeState(PlayerState.Sliding);
    }

    3 個の参照
    public override void GroundOffDown()
    {
        base.GroundOffDown();
        // 接地状態でしゃがみ解除
        // ボタンが押されていないときの処理
        ChangeState(PlayerState.Roll);
    }

    3 個の参照
    public override void AirDown()
    {
        base.AirDown();
        // スペシャル処理
        ChangeAnimation(howlHash, 0.0f);
        SEPlayer.Instance.PlaySound(SEType.Player, 4);
        Instantiate(specialObj, gameObject.transform.position, Quaternion.identity);
        playerData.ResetSpecial();
    }
}
```

ワン転（わんころ） ～ 工夫した点～

■ ステージ追加

ステージもキャラクター同様にScriptableObjectで管理しました。

ステージは一定距離進むと自動で切り替わる仕様です。

地面のマテリアル（上面、側面）や背景、敵を設定することで、ステージごとに対応したものを生成できるようにしました。

ステージ 1



ステージ 2



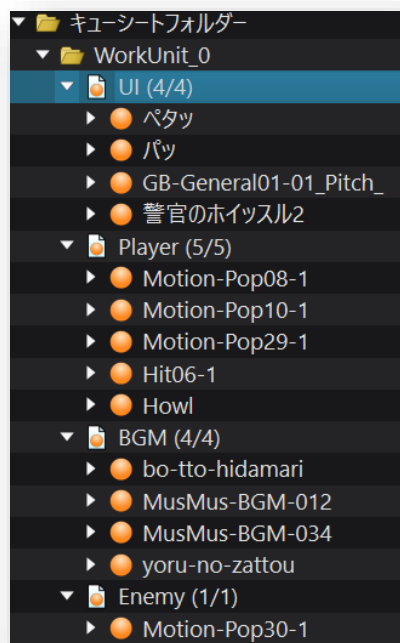
ワン転（わんころ） ～ 工夫した点～

■ サウンド管理

サウンド管理はCRIミドルウェアの「ADX LE」を使用しました。

この処理はインスタンス化したので、「using Wankoro.Sound」を追加すれば、どこからでも再生処理を使えるようにしました。

キューシートをカテゴリーごとに分けて管理しました。



SEを流すときはカテゴリーとスプレッドシートに書かれた番号を選択することで再生できます。

SE再生処理

```
/// <summary>
/// 音を鳴らす
/// </summary>
/// <param name="category"></param>
/// <param name="num">再生したい音の番号</param>
55 個の参照
public void PlaySound(SECategory category, int num)
{
    if (category != SECategory.UI)
    {
        cueSheetList.AddCueSheetList(category);
    }

    // 流すCueSheetを指定
    atomExAcb = CriAtom.GetAcb(category.ToString());
    // CueSheetの中身を取得
    cueInfo = atomExAcb.GetCueInfoList();
    // CueSheetの中身の何番目を流すかを設定
    atomExPlayer.SetCue(atomExAcb, cueInfo[num].name);
    // 音の再生開始
    atomExPlayer.Start();
}
```

スプレッドシート

番号\カテゴリー	Player	Enemy	UI
0	ジャンプ	吹っ飛ばし	画面遷移するとき
1	踏んづけ		メニューを開く、
2	クッキー取得		レベルアップ
3	ダメージ		ゲームオーバー

カテゴリー

```
public enum SECategory
{
    Player,
    Enemy,
    UI
}
```