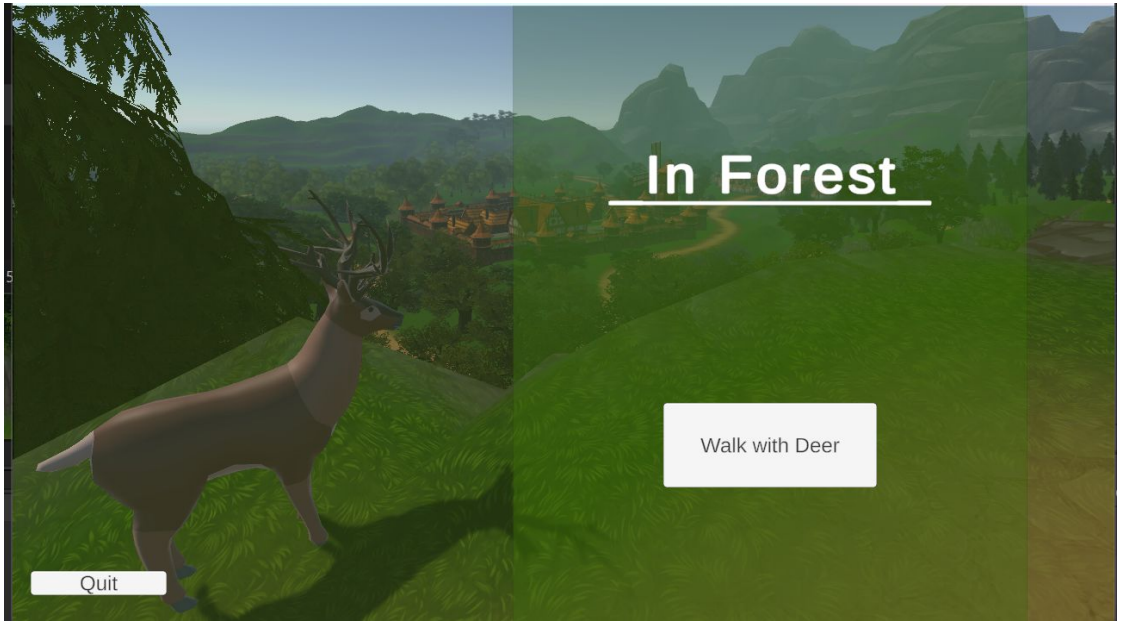
サウンド制作、音楽制作志望についてのポートフォリオ

小倉悠人

1. ゲームサウンド制作練習作品
2. 音楽・音声制作参考作品
3. その他音に関する来歴について

1.　ゲームサウンド制作練習作品



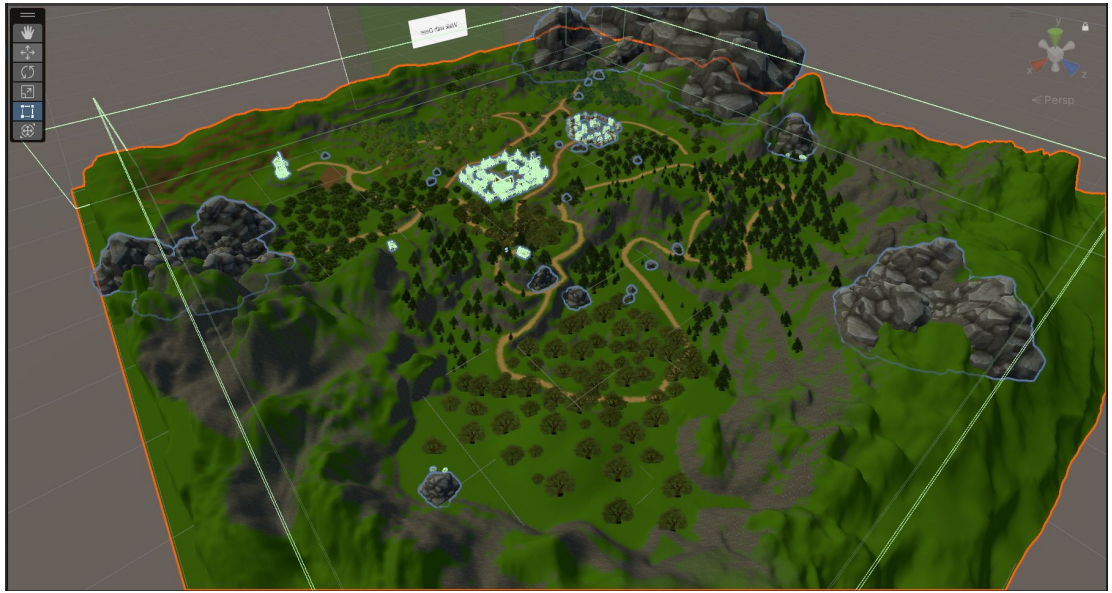
Unity + CRI ADX2練習作品

“In Forest”

職業訓練の中で学んだUnity
C#プログラミングと、
独学のCRI ADX2LE
を使用して作成しました。

森のマップを環境音を
聞きながら鹿で探索する
作品です。

GitHub : https://github.com/yuraklone/InForest



今回マップは無料アセットの
Fantasy landscapeの
デモマップを使用しました。

森だけではなく街もあるため
他の人間なども設置して
いきたかったところですが、
本来の目的である音を作成、実
装するところまでに時間が
かかる為今回はプレイヤーの鹿
が一匹歩くのみです。

FootstepPlayer.cs

Assembly-CSharp

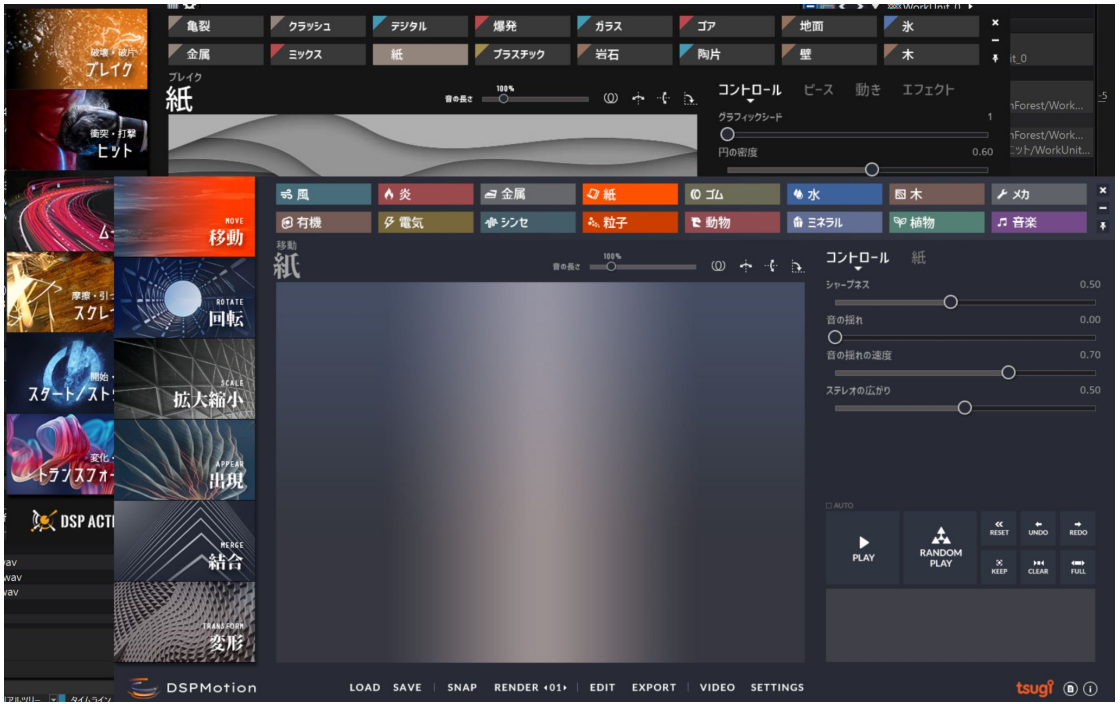
```
                switch(maxIndex)
                {
                    case 0:
                        criAtomSource.cueName = "deerfoot_glass";
                        break;
                    case 1:
                        criAtomSource.cueName = "deerfoot_cray";
                        break;
                    case 2:
                        criAtomSource.cueName = "deerfoot_rock";
                        break;
                    case 3:
                        criAtomSource.cueName = "deerfoot_cray";
                        break;
                    case 4:
                        criAtomSource.cueName = "deerfoot_rock";
                        break;
                    default:

                        break;
                    }
                break;

            case "WoodFloor":
                criAtomSource.cueName = "deerfoot_wood";
                break;
            case "Rock":
                criAtomSource.cueName = "deerfoot_rock";
                break;
```

121 % 問題は見つかりませんでした 行: 1 文字: 1 SPC CRLF

鹿の足音の実装について

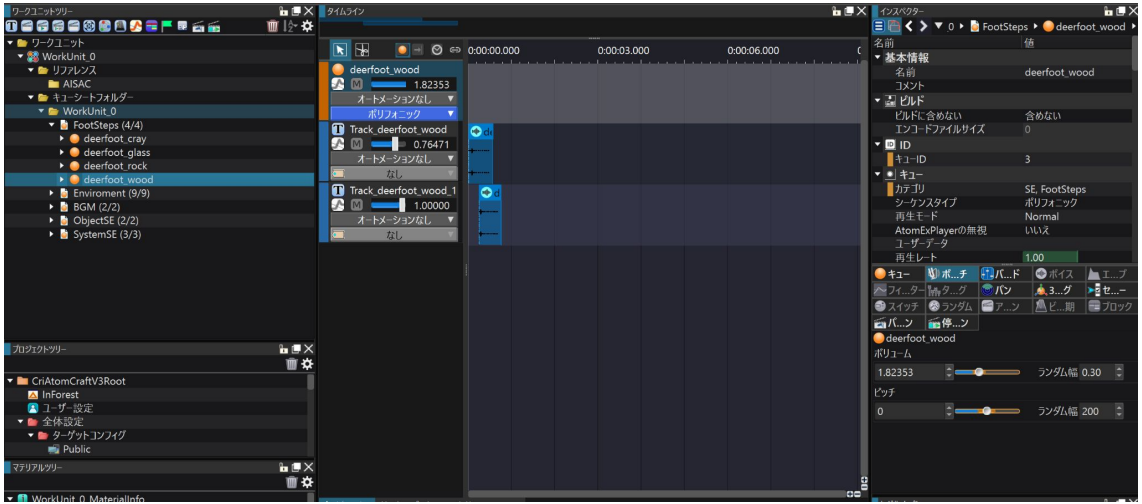
まず、Terrainによる地面の
材質の違いや、木の建物や岩
など地面以外の乗って歩ける
オブジェクトも多数あるため
タグとTerrainレイヤーから
AtomSourceのキュー名を
切り替えるスクリプト作成、

このキュー名通りにキューを
AtomCraft側で作成しました。



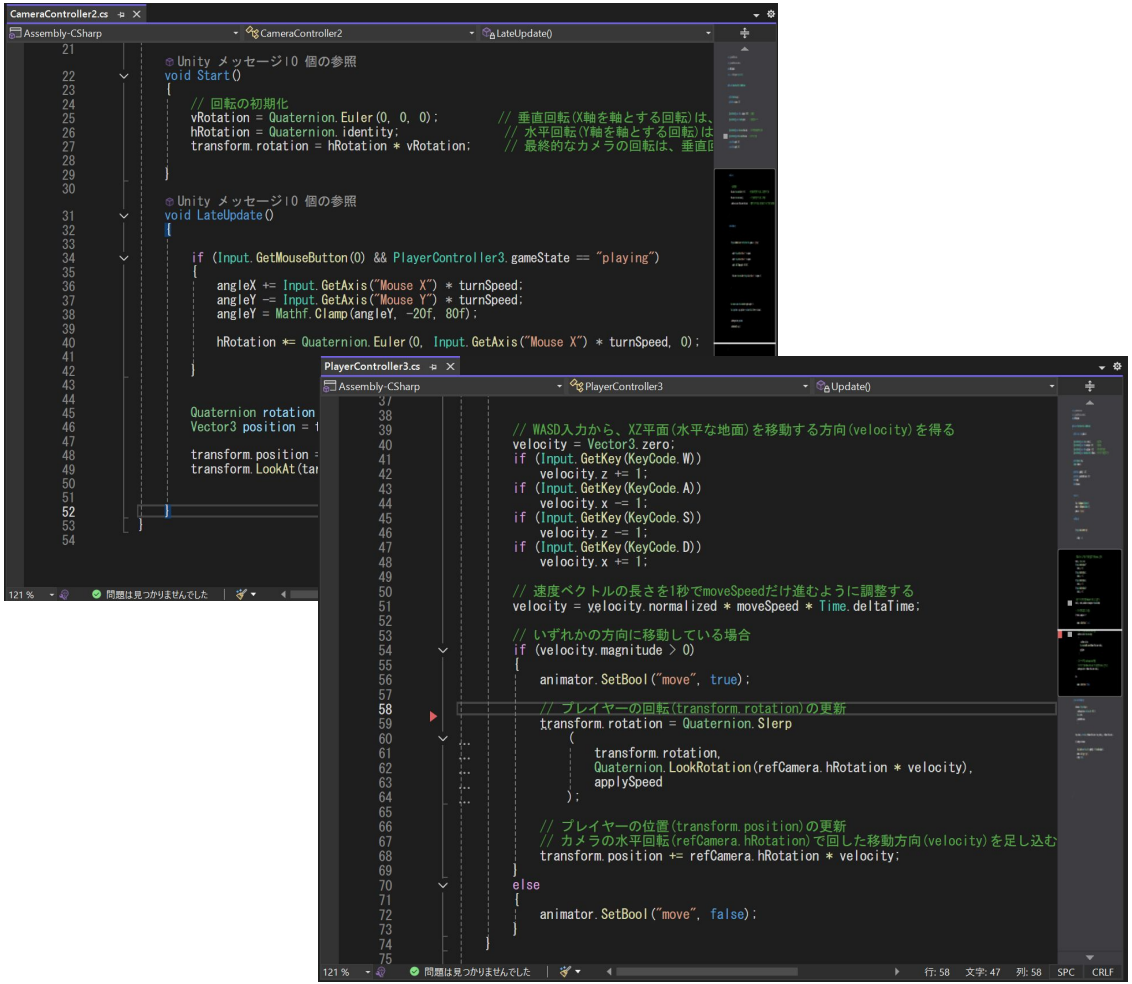
足音および以後の効果音作成には基本的にTsugi Studio社
『DSP Action』
『DSP Motion』
『DSP Fantasy』
を使用しています。

また後述の環境音では鳥の声等で一部(株)プラスシグナル社の効果音ライブラリを使用していますが、基本的に上記3つで作成しました。



DSP3種は名称に近しきものはあるものの、それらではイメージと遠く、尚且つ今回歩くのは鹿なので、

各種実際に音を聞きながら音を作成し、キュー一発につき前足と後足の2着地分鳴るように音素材もしくはキュー上で作成しています。



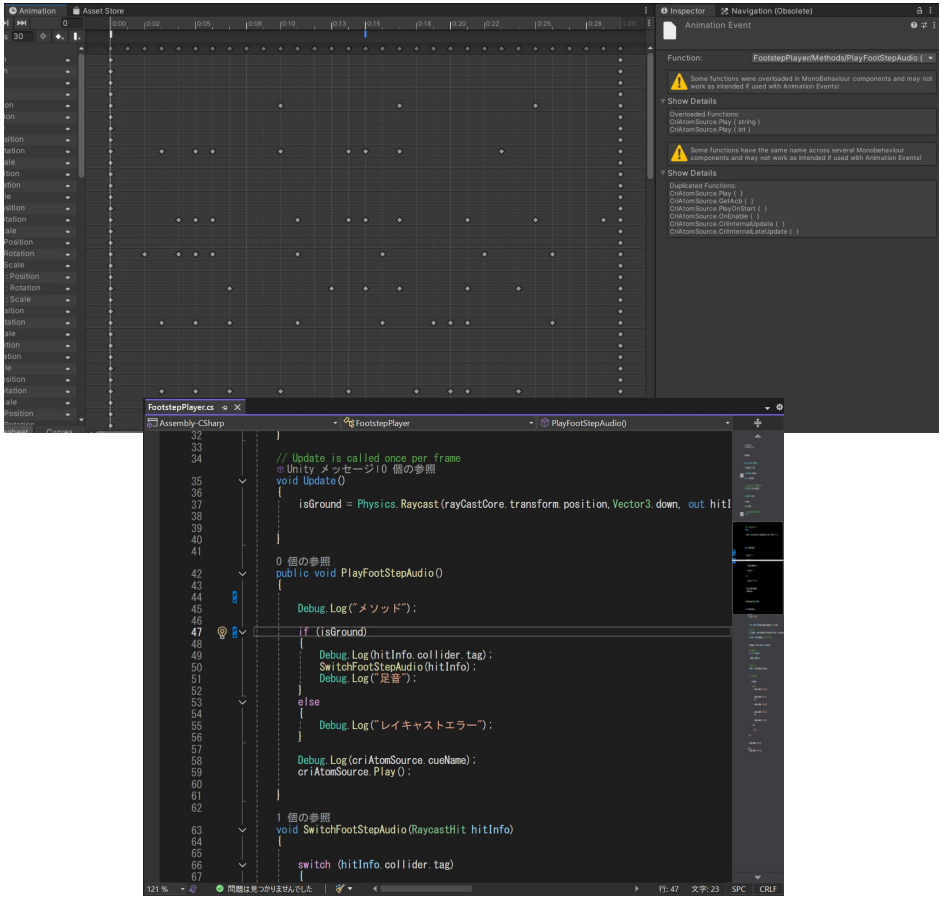
```
CameraController2.cs
Assembly-CSharp | CameraController2 | LateUpdate()

    Unity メッセージ | 0 個の参照
    void Start()
    {
        // 回転の初期化
        vRotation = Quaternion.Euler(0, 0, 0);          // 垂直回転(X軸を軸とする回転)は、
        hRotation = Quaternion.identity;                // 水平回転(Y軸を軸とする回転)は
        transform.rotation = hRotation * vRotation;     // 最終的なカメラの回転は、垂直回

    }

    Unity メッセージ | 0 個の参照
    void LateUpdate()
    {

        if (Input.GetMouseButton(0) && PlayerController3.gameState == "playing")
        {
            angleX += Input.GetAxis("Mouse X") * turnSpeed;
            angleY -= Input.GetAxis("Mouse Y") * turnSpeed;
            angleY = Mathf.Clamp(angleY, -20f, 80f);

            hRotation *= Quaternion.Euler(0, Input.GetAxis("Mouse X") * turnSpeed, 0);

        }


        Quaternion rotation
        Vector3 position = t

        transform.position =
        transform.LookAt(tar

    }
}

121 %   問題は見つかりませんでした
```

```
PlayerController3.cs
Assembly-CSharp | PlayerController3 | Update()

        // WASD入力から、XZ平面(水平な地面)を移動する方向(velocity)を得る
        velocity = Vector3.zero;
        if (Input.GetKey(KeyCode.W))
            velocity.z += 1;
        if (Input.GetKey(KeyCode.A))
            velocity.x -= 1;
        if (Input.GetKey(KeyCode.S))
            velocity.z -= 1;
        if (Input.GetKey(KeyCode.D))
            velocity.x += 1;

        // 速度ベクトルの長さを1秒でmoveSpeedだけ進むように調整する
        velocity = velocity.normalized * moveSpeed * Time.deltaTime;

        // いずれかの方向に移動している場合
        if (velocity.magnitude > 0)
        {
            animator.SetBool("move", true);

            // プレイヤーの回転(transform.rotation)の更新
            transform.rotation = Quaternion.Slerp
                (
                    transform.rotation,
                    Quaternion.LookRotation(refCamera.hRotation * velocity),
                    applySpeed
                );

            // プレイヤーの位置(transform.position)の更新
            // カメラの水平回転(refCamera.hRotation)で回した移動方向(velocity)を足し込む
            transform.position += refCamera.hRotation * velocity;
        }
        else
        {
            animator.SetBool("move", false);
        }
    }

121 %   問題は見つかりませんでした   行: 58   文字: 47   列: 58   SPC   CRLF
```

やや余談にもありますが、
今回プログラミングの練習も
兼ねてよくある三人称視点の
操作で鹿を移動させています。

WASDでカメラ内の上下左右、
マウスで視点操作の参考は
完全な狙い通りのものは
無かったため、今回複数の例を
参考に理想形を組みました。

足音をAnimation内で発動
するため、Animatorの遷移も
ここで制御しています。



足音の発音タイミングは
Animationからメソッドを
呼び出し、AtomSourceから
発音させています。

このスクリプトだけDebug.Log
が多いですが、Raycastでの
判定と発音に苦戦した名残で、
この鹿自体が原点が地面と
なっていたため、原点から
Raycastを伸ばすのでは
ちゃんと判定しなかったため、
Raycastを伸ばす位置の調整で
解決しました。

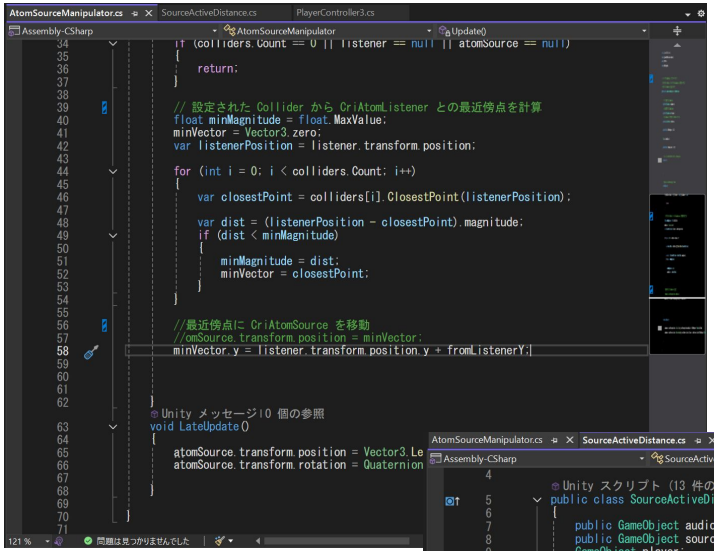
環境音の実装について

次に、森を探索するため、
木々が風でざわめく音や、
鳥や虫の鳴く音を設置します。

UnityとCRIだけでは範囲での
発音は出来ないため、CRIの
公式のチュートリアルを参考に
範囲のコライダーとその中を
追従する音源を実装しました。

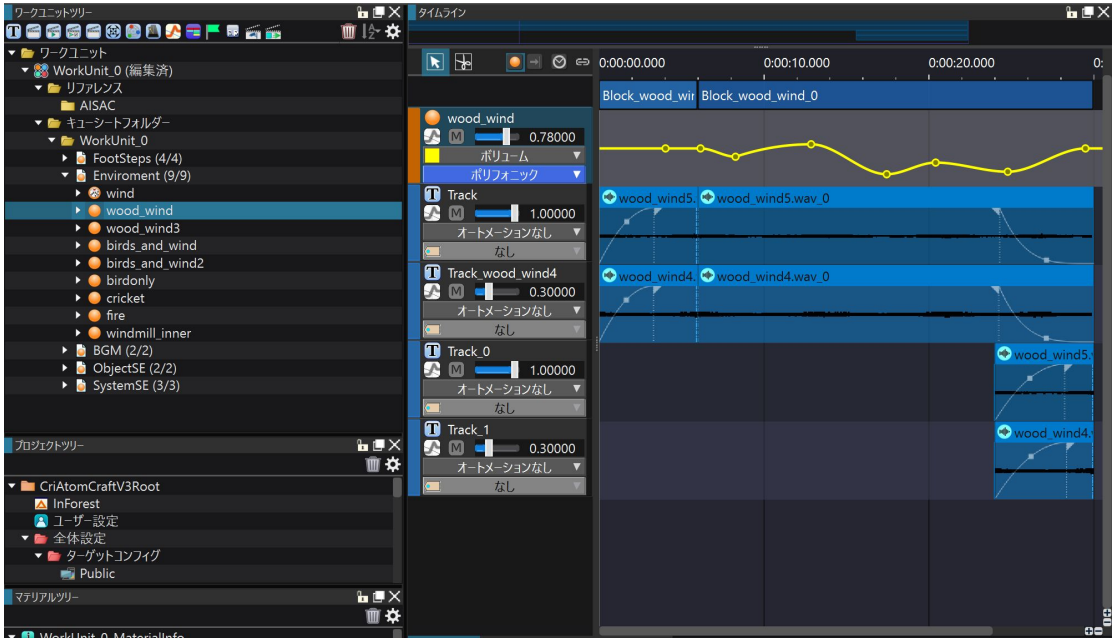
今回AtomListenerの設置位置としては、三人称視点故にカメラだと3Dポジショニングを適用してしまうと動きに対してパンニングが過度に暴れてしまうのと、かといって鹿視点ではやや違和感があったため、鹿よりやや上に設置しています。

また、回転を完全に追従してしまうとこれもパンニングが暴れすぎてしまうため、プレイヤー回転より線形補間でやや遅く回るようにしました。



また、コライダー内の音源側のプレイヤーへの追従も同様に、Updateに直接放り込んでしまうと微細な動きでもパンニングが暴れまわってしまったため、こちらも追従は線形補間でスピードを制御し、パンニングの暴れを緩和させています。

また、これらが動き回るのも鳴ったままも余分な負荷と発音数を食うため、別途でコライダーとの距離で音源のON/OFFを制御しています。

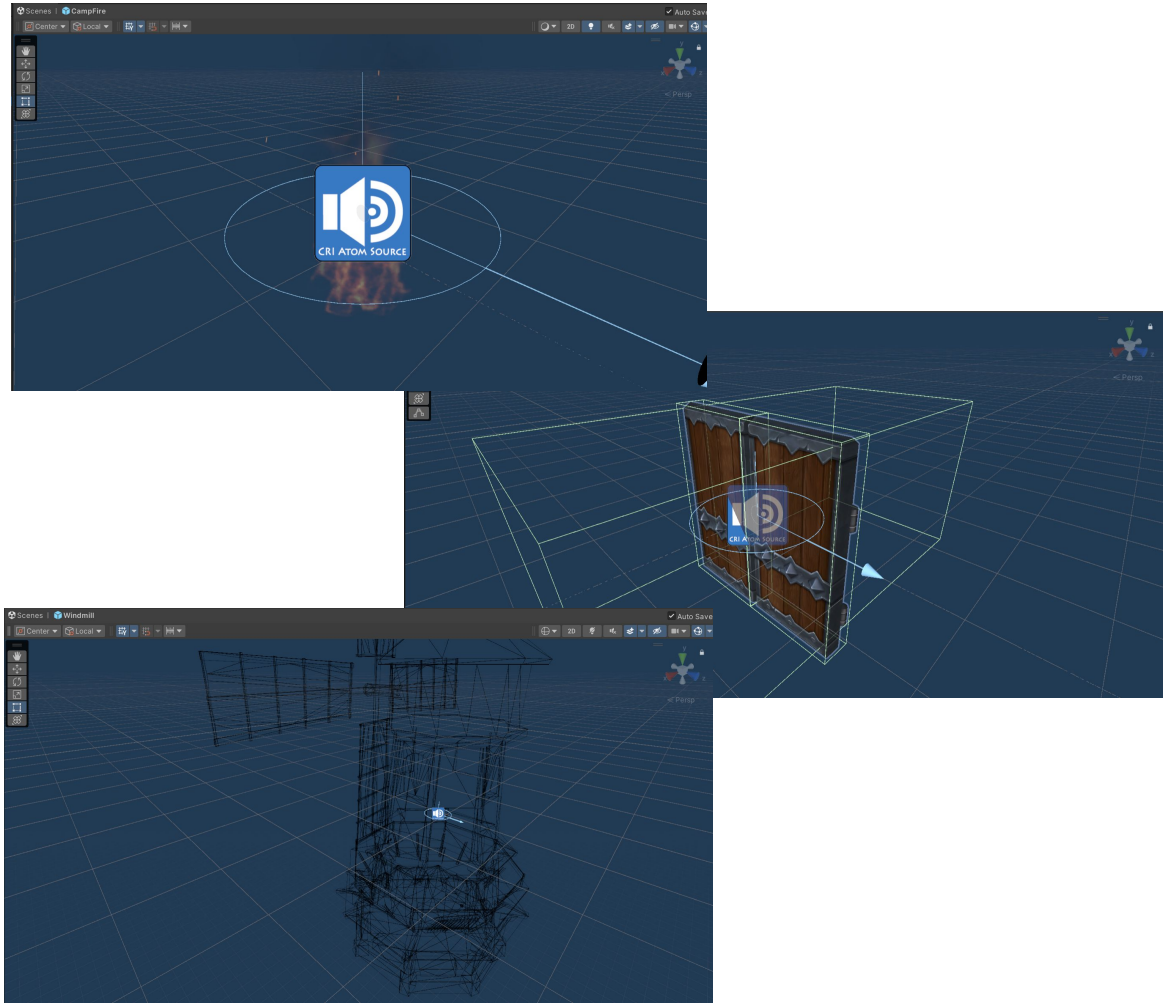


先述の音源から鳴らす環境音はDSP3種から主に作成し、作り切れなかった鳥の鳴き声と虫の鳴き声のループは以前購入したプラスシグナル社の効果音ライブラリを利用させて頂きました。

本来サウンド志望であればフォーリー録音やフィールドレコーディングへ赴くべきと承知しておりますが、その時間と機会、機材もない中での制作だったため、実装とAtomCraftでの実践を重視しました。

エリア内で音源たちはこの様に配置され、Listenerの上側が風やざわめき、鳥の声などを発音し、下側が虫の声、鹿の足元は先述の足音となりパンニング不適用です。

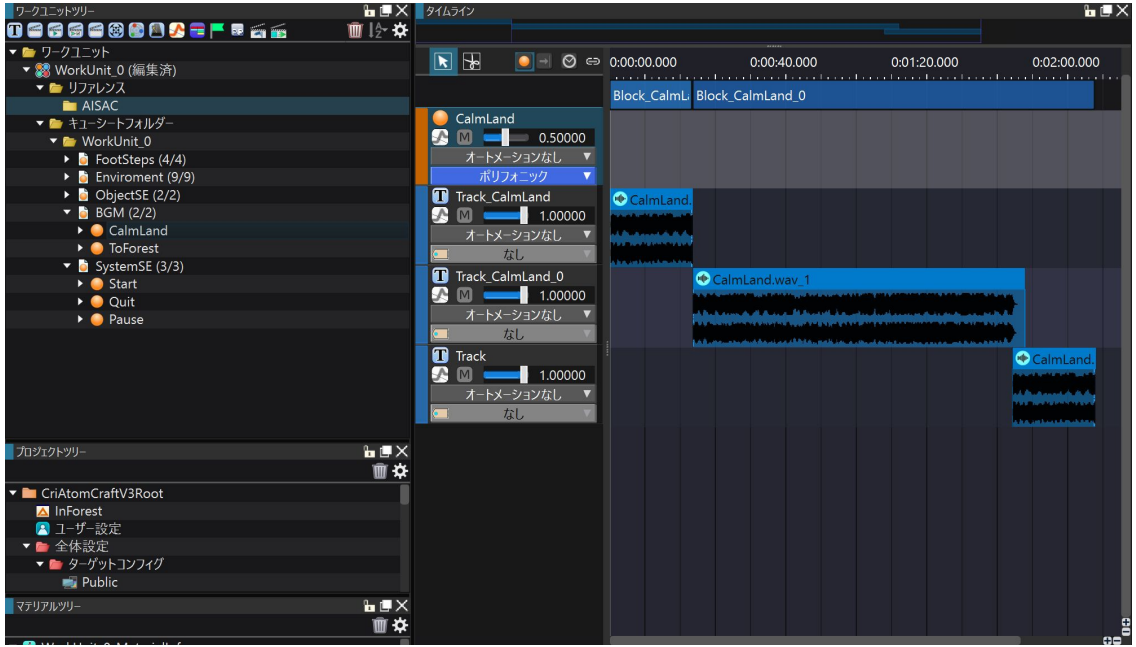
当初リアリティから上下により距離を置いた追従にしていたのですが、その場合左右のずれがよりシビアになってしまい、演算の誤差かじっとしていても勝手に片側に寄ってしまうなどのバグが出たため、それがない距離で上下差を出しています。



その他焚火や各家の中の暖炉の炎、木製のドア、風車の内部の作動音などはPrefabに直接配置して発音させています。

これらも鳴ったままでは負荷と余分な発音数となってしまうので、距離によるON/OFFを再生/停止を適用しています。

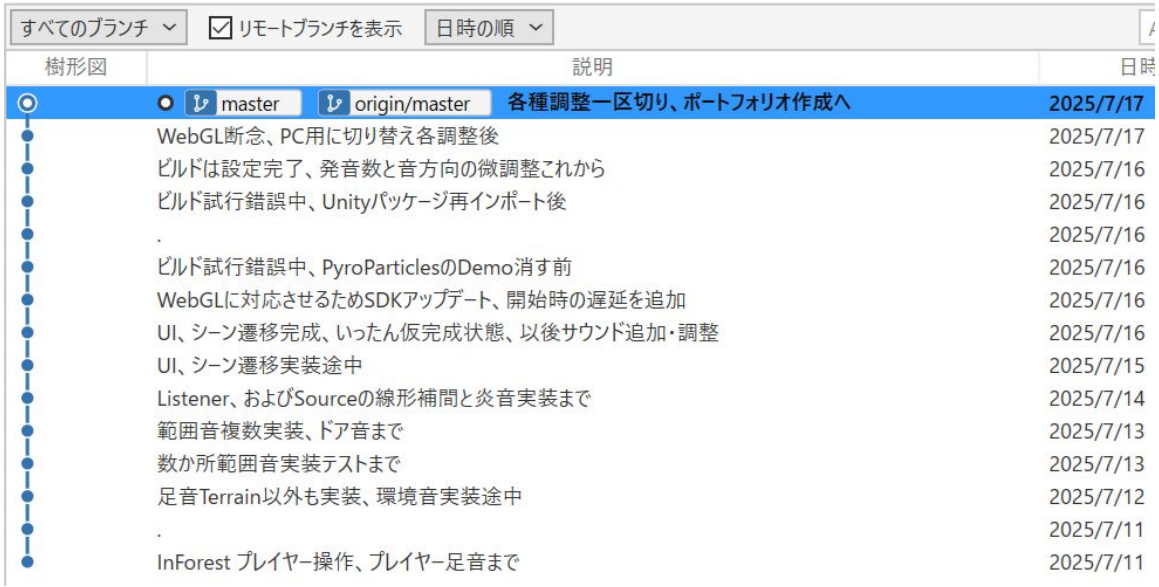
AtomCraft側の各発音数制限だけでは、うまく発音と停止を制御しきれなかったため、Unity側でも対策した形です。



BGMとSEの実装について

BGMに関しては元々作曲が本来やってきていたことのため本作くらいのループBGMであればタイトル、ゲーム内各数時間程度で作曲できます。

また各ボタン押下時のSEに関しては今回鹿の鳴き声のような効果音を作成し、短い環境音とともに各種キューを作成しました。



制作期間

訓練以外の時間から制作を始め訓練内での作業解放も含めておおよそ7日間ほどでの作業となりました。

試行錯誤やバグ解決も多かったため要所要所で時間はかかりましたが、事前のCRI ADX2 LEについての学習や、音楽制作で使ってきた音加工の経験から、実際音に関わる部分では効率的に進められたかと思います。

2.　音楽･音声制作参考作品

以下リンクにて参考作品を載せております。
比較的ジャンルの幅広い作曲ができること、またこれらの参考程度の
楽曲であれば数時間～数日(1日8時間労働換算)で作曲できる速度は
自負しています。

練習制作

(※最近の音源のフォーマットが mp3となっているのは Google側の容量が厳しくなってきたためです、
またブラウザ上で再生すると非常に音質が悪くなります、ミックスは比較的ラフに近いですがそこまで
酷い音質には仕上げておりません。)

音楽制作、加えてシンセサイザーを使用するSE制作などは
慣れているのと試行錯誤の速度重視で主にLogic Proで行っています。
但し、過去ProToolsやSamplitudeも使用し、効果音の調整や加工には
SOUND FORGEも扱った経験があります。

3.　その他音に関する来歴について

1993年9月19日生まれ
・2歳からヤマハ音楽教室に通い始め、エレクトーンを専攻。
　エレクトーン専攻後は演奏の練習と同等に迫る時間を音作りに費やすような
　幼少期を過ごす。

・茨城大学工学部へ進学、電磁気学と情報工学をいずれも学ぶ学科で、
　卒業研究時は物体の振動と信号解析を専門とするゆらぎ・雑音工学研究室へ所属。

・同時に大学時代はバンド活動を元に自宅録音によりレコーディング技術の
　基礎を独学で習得。ソフト・ハード面いずれにおいても音響処理への理解を深める。

・新卒時にゲームサウンド職への就職が叶わず、株式会社小柳出電気商会へ就職。
　大学での電磁気学と情報通信の分野での内容を活かし、ミュージシャンや
　サウンドエンジニアの方々の相談、サポートを業務内で担当。

・次の転職先FTF株式会社ではオーディオ機器全般の担当としてジャンル問わず音響機器に関する
　社内業務全般を遂行。

・その他上記2社へ所属している間も作曲やレコーディングに関して独学で学習を続ける。
　※7/21現在サウンドレコーディング技術認定試験受験後結果待ち