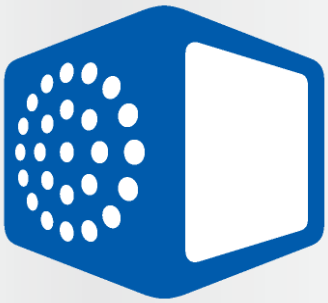


CRIWARE®

UE4×ADX2
Beginners Book

UE4×ADX2 導入マニュアル

UNREAL ENGINE

ゲームなどのエンターテイメントにおいて、Unreal Engine で制作されたアプリケーションでは、
よりリアルで、ハイクオリティなビジュアル表現が使われるようになりました。

進化するビジュアル的な演出に合わせて、BGM やエフェクト音などサウンドの付与も重要視され、
取り扱うサウンドの数も膨大な数になってきてきたため、昨今ではすべてのサウンドをゼロから作るのではなく、
「いかに効率よく」一つのサウンドを「バリエーション豊かに」再生できるかが重要となってきています。

本書は、そうしたバリエーション豊かなサウンド再生についての入り口にたてるよう、
どのように考えて ADX2 を使って実装していくのかを事例を交えて紹介していきます。

Topics

chapter 1 ADX2 を用いたサウンドデザインを行う前に

本章では、UE4 や ADX2 に触れる前に、どのようにゲームのサウンドを
デザインしていくのかを、実装手順を交えて説明していきます。

そもそもゲームサウンドって？

まずゲームサウンドには下記のようなものがあります。

・BGM
・キャラクターボイス
・UI 音
・キャラクター動作音
・エフェクト音
・環境音

キャラクターモデリング、アニメーション制作、ビジュアルエフェクトなどは個別の仕事として分業化が
行われていたりしますが、ことサウンドについては、UI、キャラクター、環境、などなど……
ゲーム全般の仕様を踏まえた上でサウンドの付与を行っていく必要があります。
UI がどのように遷移していくのか、キャラクターアニメーションがどのように繋がっていくのか、
どのエリアが洞窟で、どのエリアが草原か、など。
サウンドの種類によって気にしなければならない点も異なってきます。
そして、ゲームの規模が大きくなればなるほど、制作するサウンドの量も膨大となってきます。

環境音の場合で見てみましょう。
森のシーンであれば環境音として再生するなら、簡単に上げるだけでも、鳥の声、風音、雨音、虫の声などがあります。
天気の良いときは鳥の声や虫の声はにぎやかで、風の音は静か、雨音はしないものです。

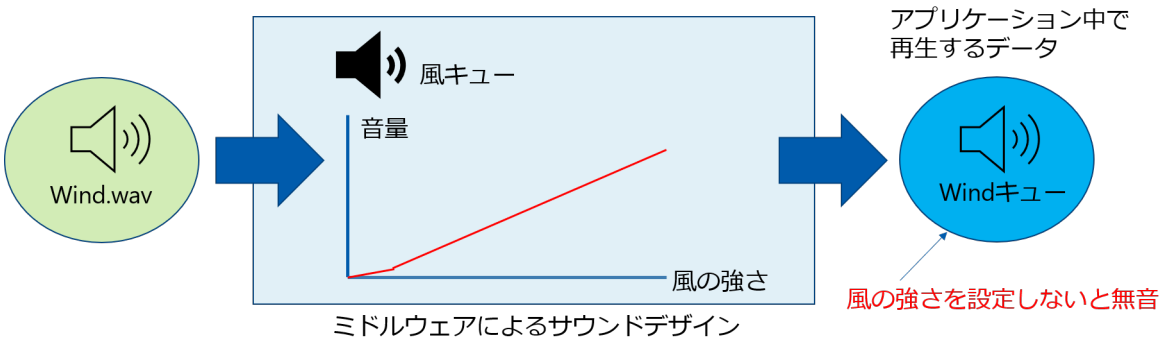
しかし、天気が悪くなると、その度合いに応じて鳥の声や虫の声は静かになっていき、
風音や雨音などは強くなっていきます。

また、敵キャラクターが出てきて戦闘などになると、鳥や虫が逃げ出そうとしたりするため、
戦闘時、非戦闘時においても環境音のにぎやかさが変わっていきます。

これらの状況に応じてサウンドデータをひとつずつ作成していくことも可能ですが、
プロジェクトが大規模になればなるほど、作業量やデータ量が増えていくため、現実的ではありません。

では実際にどうやっているかというと、サウンドのボリューム値やピッチ値の変更、エフェクトの適用などを行い、
鳴り方にバリエーションを持たせ、シーンに応じて鳴らしています。

こうした設定を付与したサウンドデータには一つ一つのサウンドに用途や仕様が含まれるようになるため、
別のシーンで同じ波形を使おうとすると、まったく違う鳴り方になってしまう、といった思わぬ落とし穴になることもあります。



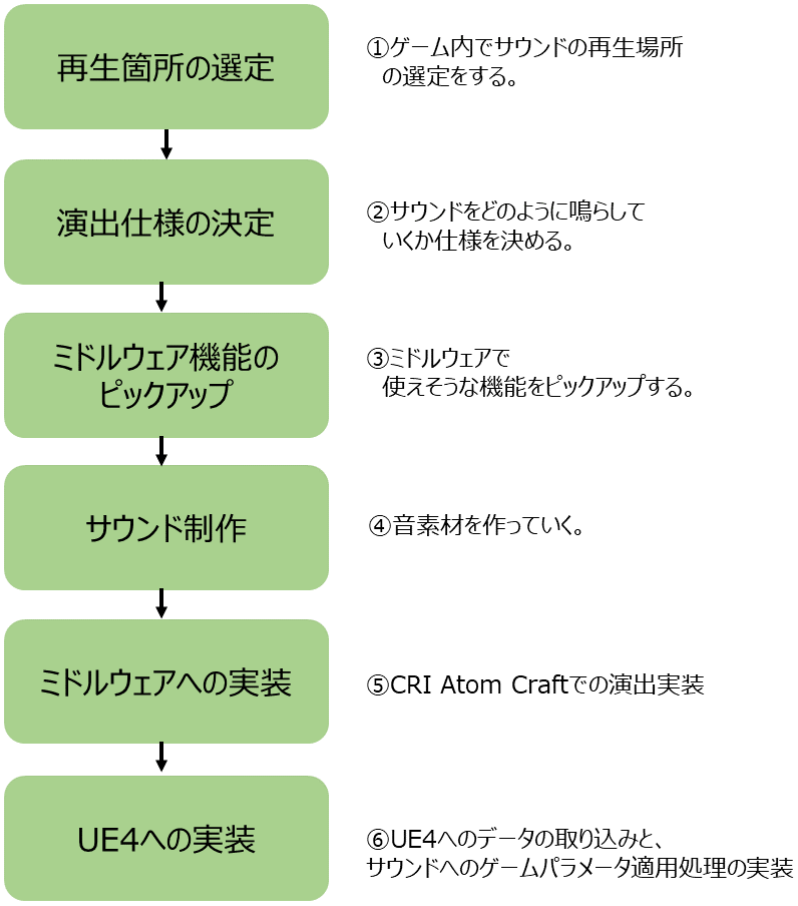
風の音を例に取り上げてみましょう。
この図においては出力する「Wind キュー」はもともとは「Wind.wav」という波形でした。
しかし、「風の強さパラメータ」によってボリュームが変わるように設定した場合、
単純に再生した場合は風の強さパラメータが「０」として再生され、無音になってしまいます。

つまり、このサウンドをほかの場所で流用しようなどした場合、風パラメータの存在がない場合には
常に無音となり、再生はされているのに音が鳴らない、というようなケースに陥ることがあります。
そのため、ゲーム向けにデザインされたサウンドは仕様・用途があることに十分気を付けていく必要があります。

実装におけるワークフロー

次にゲームサウンドのデザインフローを解説していきます。

基本的に単純なワンショット再生でも事足りますが、
前項目で説明したようにゲーム中のパラメータを使って再生にバリエーションを持たせたい場合は
次のようにデザインしていくことをお勧めします。



このように演出仕様を決めて、そこからどのようにアプリケーション中に実装していくかを、
エンジンやミドルウェアの機能から決定していきます。
実装仕様をある程度見定めることで、どういったパラメータが必要か、などがイメージできるようになり、
サウンド演出実装からアプリケーション組み込みまでのフローがスムーズになりやすいです。

ゲームサウンドをデザインするうえで気にする必要があること

音の再生方法、鳴らし方の設定は ADX2 のサウンドオーサリングツールである CRI Atom Craft と UE4Editor の両方で行っていくことになります。

ただ、CRI Atom Craft でサウンドデザインを行う前にどのようなサウンドが必要か、
どのような仕様で実装していくのかを事前に煮詰めていく必要があります。
サウンドの仕様のうち、どこまでが CRI Atom Craft で実装するべき内容か、
どこまでが UE4Editor 中で実装していくかを事前に見極めていくことが重要になります。

サウンド実装はゲーム全般の体験にかかわってくるため、サウンドデザイナーはプランナーやディレクターと一緒にゲームの仕様書を確認したり、ある程度出来上がっているプロジェクトを確認したりして、
サウンドが必要な場所をリストアップしていき、どのようなサウンドを作っていかなければいけないのかを明確にしておく必要があります。
これを踏まえて、ゲーム中からどのようなパラメータを受け取って音を変えていくかを想定しながら、
サウンドデザイナーは CRI Atom Craft でサウンドデザインをしていくことになります。

■ CRI Atom Craft での作業
CRI Atom Craft では、足音の指定や、サウンドのカテゴライズや鳴らしわけなど音そのものに対してのデザインを行っていきます。

・足音（水、鉄、草で切り替えられる）
・時間によって変化する BGM
・建物内外での音の響かせ方
・セリフ中は BGM を小さくするという設定
・最大発音数
・サウンドのカテゴライズ（BGM、SE、CHR）
・3D サウンドの減衰設定
…etc

CRI Atom Craft でのデザインが終了したら、
次は作成したデータを UE4 に取り込んで実際に再生できるようにする作業になります。

■ UE4Editor での作業
UE4 側での操作については CRI Atom Craft でデザインしたサウンドデータを取り込んで、ゲーム中の状況をパラメータとしてサウンドに適用しつつ再生するだけです。

・ゲームの時間（ゲーム内の朝昼夜）
・プレイヤーの移動速度
・プレイヤーの環境（地面が鉄とか、建物の中にいるとか）
・タイトル画面、メニュー画面、ゲーム画面への移動
…etc

作業としては、上記のようなゲーム中の変化を、パラメータとして Blueprint などで ADX2 に情報を渡します。

■アプリケーション中での実装作業

この時、パラメータは CRI Atom Craft でデザインした際にデータ側で用意したセレクタラベル名や、
AISAC Control パラメータなど ADX2 専用のパラメータを変更していくことになります。
そのため、サウンドデザイナーはプログラマと ADX2 で設定したパラメータの共有をしておく必要が出てきます。

例えば、雨音のサウンドを作ろうとした際に再生条件に下記のようなものを追加したいとします。

・雨音は一つのサウンド
・雨音の強弱をコントロールできる
・屋根のある建物の中に入った場合に音がこもるようにする

こういった条件をサウンドデザイナーやプランナーが上げた場合に、プログラマーはこの条件を
実現していかなければなりません。
この時にサウンドデザイナー側とプログラマ側で情報の摺合せが必要となってきます。

その場合上の条件でプログラマが必要とするものは、

・雨音は一つのサウンド
・雨音の強弱をコントロールできる
　→パラメータでコントロールできる
・屋根のある建物の中に入った場合に音がこもるようにする
　→遮蔽判定のフラグの用意が必要、遮蔽具合をパラメータでコントロールできる

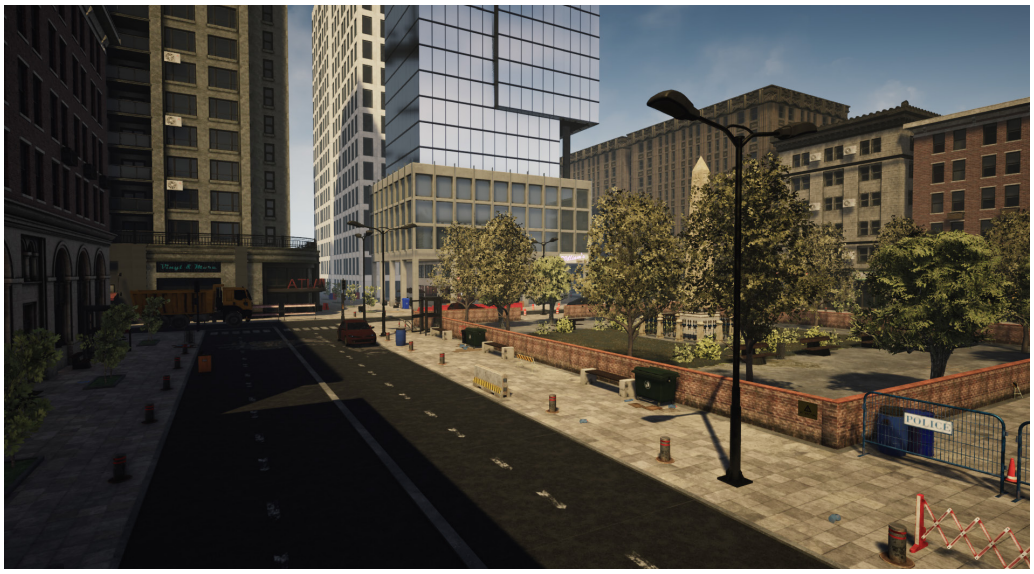
という、数値的に変えるものがどの部分にあたるのかということになります。
そのため、サウンドデザイナーやプランナーの方はどの部分をパラメータで変更するのかを明確にして、
プログラマに演出したい内容とその演出に必要なパラメータについて共有することが重要になります。

chapter 2 アンビエントサウンド

この章では前章の内容を踏まえ、ADX2 の機能を使ってアンビエントサウンドをどのように作成していくかを実例ベースで解説していきます。

アンビエントサウンドとは？

昨今では広大なフィールドを舞台にしたゲームが多数パブリッシュされています。



そうしたゲームの舞台では、草木が生い茂るような世界だったり、近代的な建物が並ぶ世界だったり、空気が薄い星が舞台だったりと様々です。

アンビエントサウンドとはそうした環境を彩るためのサウンドになります。

森が舞台のシーンでは木々のざわめき、風の音、虫や鳥の声などが挙げられますし、洞窟では、風の音、鍾乳石から水が滴る音、石が落ちる音、雨音などが挙げられます。

こうした音は、パーティクルなどのエフェクトと一緒に空間全体で鳴っているように再生することになります。

シンプルなアンビエントサウンドとその課題

ではアンビエントサウンドはどのように再生するのでしょうか？
分かりやすい例としては、空間全体でループ音を再生し続けるというものです。
鳥の鳴き声、風の音、虫の鳴き声等を一つの長尺のサウンドとして収録し、ループ音として再生するというものです。

これでもある程度は環境を彩ることができますが、もっと演出上良くなるポイントを挙げることができます。

しかし、ここでいくつか改善したらもっと演出上良くなるポイントが挙ることができます。

■改善したほうが良い問題点
・ループ音の場合、どんなに長い尺で収録した音であってもループ音ということで、
　音のなり方に規則性が出てきてしまう。

・長尺の音ということで、不自然差が感じられない程度の長さのループ音を作ってしまった場合、
　一つの音のデータ量がその分大きくなってしまう。

より有用なアンビエントサウンド

ではより有用なアンビエントサウンドとは何でしょうか？
上記で挙げた方法はループ音での演出でした。
ループ音を使った場合の問題点は先述したものを簡単にすると次の通りです。

・音の繰り返しになるため規則性が出てしまう。
・長尺のサウンドを利用する場合はデータ量が増えてしまう。

このデメリットを解消した場合、次のような環境音になってきます。

・音の再生に規則性がなく不連続
・停止するまで再生が続く
・データ量が小さい

より有用なアンビエントサウンドを目指す場合にはこの３点を目指していくことをお勧めします。
先のデメリットの２点目の「長尺のサウンドを利用する場合はデータ量が増えてしまう」問題は、
規則性を隠すために長尺のサウンドを準備しなければならなかったというのが主な原因です。
これに対して、そもそも別の方法で無限長のサウンド再生を実現できれば問題がありません。
その方法として、一つのループ音を再生しつつ、装飾音としてランダムな単発音を、
ランダムなタイミングで鳴らし続けるという方法で対処できます。

また、先に述べた目指すべき条件に加えて、

・ゲームの状況に応じてサウンドの鳴り方が動的に変化する

という演出を加えることで、ゲーム世界の彩りをさらに持たせることができるようになってきます。
これは、天気の状況やゲームの進行度、Player のライフ等のパラメータをサウンドに紐づけることで、
環境音を周囲の状況に応じてにぎやかにしたり静かにしたりというような演出を作っていくことができます。

パラメータを徐々に変化させていくことでサウンドも徐々に変化させていくというのは、
瞬間的に環境音が変わってしまうことに比べ、ユーザーが気づきにくいものになってきます。

ですが、この「変化が気付きにくい」ということが体験デザインにおいては非常に重要です。

ユーザーが体験に没入している状況下では、体験中の気持ちの遷移を途切れさせないため、
サウンドの再生を「ゲームの状況に応じてサウンドのなり方が動的に変化する」よう、
ゲームの遷移を意識したうえで一歩進んだサウンドデザインをしていく必要があります。

プロシージャルデザイン

サウンドのプロシージャルデザインとは、細かい単位で処理を行った複数のサウンドを、
一つのサウンドにまとめて再生できるようにしていく、というサウンドデザイン手法です。

例えば、「鳥の鳴き声」のような環境音をデザインしていくときにこのデザイン方法が役立ちます。

『鳥の鳴き声』
鳥の鳴き声の要素を簡単にまとめていくと次のような形になります
（説明を簡単にするため少ない要素にしています）。

・ずっと発音され続ける
・複数回同時に発音される
・不規則なタイミングで発音される
・発音の度異なる音色になる

プロシージャルデザインは、この小さい要素を組み合わせていくことで
鳥の鳴き声を環境音として作っていくことができます。

順序としてはまず「発音の度異なる音色となる」鳥の声サウンドを作成します。

１.「発音の度異なる音色となる」
　短い鳥の鳴き声の波形を複数用意して、ランダムに再生する仕組みを作成する。
次に「発音の度異なる音色となる」サウンドを不規則なタイミングで発音するようしていきます。

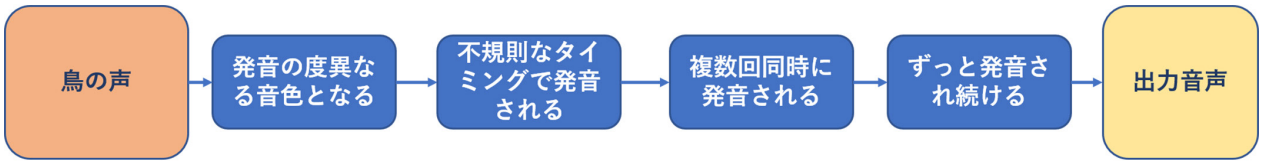
２.「発音の度毎回異なる音色となる」サウンドを「不規則なタイミングで発音される」ようにする。
　再生の度にランダムディレイを行うようにする。

次に「複数回同時に発音される」ようにするため、２で作成した再生フローを複数回同時にトリガーされるようにします。

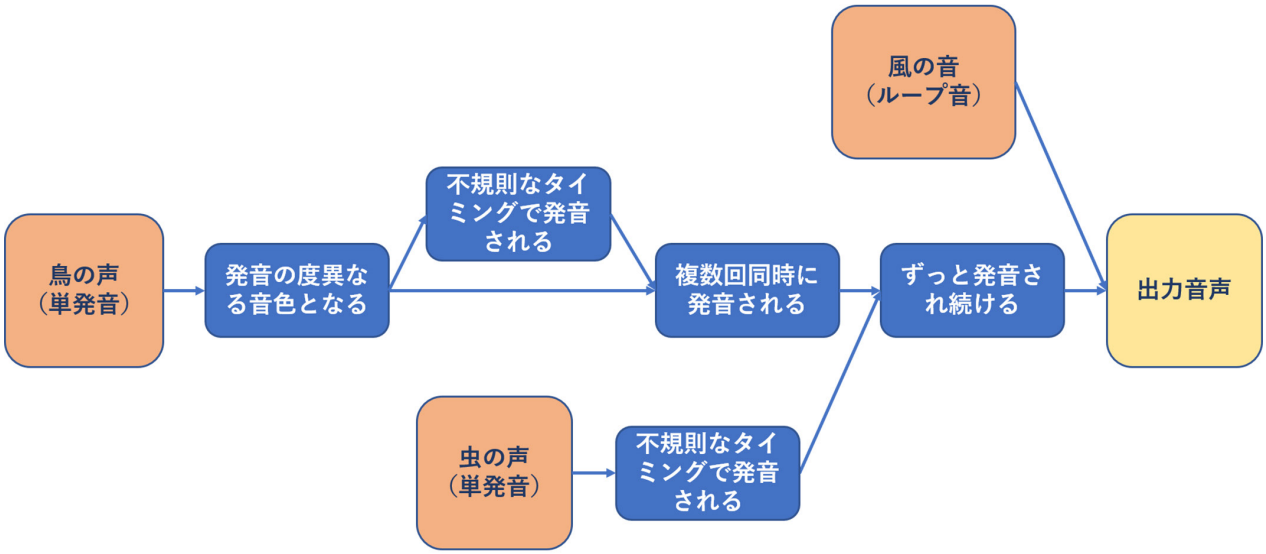
３.「発音の度異なる音色となる」サウンドを「不規則なタイミング（で発音される）」で「複数回同時に発音される」ようにする。
トリガーされる回数をランダムに設定し、２までのフローが複数回同時に行われるようにする。

そして、「ずっと発音され続ける」よう再生が終了したら、再度３までのフローがトリガーされるようにします。

４.「発音の度異なる音色となる」サウンドを「不規則なタイミング（で発音される）」で「複数回同時に発音される」ようにしたものを「ずっと発音され続ける」ようにする。
３までのフローによる発音が停止したら、再度３までのフローを繰り返す。



このように小さな機能を作っていったら、その機能を内包して新しい機能を継ぎ足していくデザインの方法をプロシージャルデザインと呼びます。



説明では一つのフローだけご紹介しましたが、処理の途中で別のフローと合流して一つの流れになったり、
一度分岐した後にまた本流に合流したりして一つの流れに戻る、といったように組んでいくのが
プロシージャルデザインの本質です。

一見文章上ではあまりイメージがつきにくいと思いますので、
後述する CRI Atom Craft でのサウンドデザインでプロシージャルデザインを実践していきます。

この手法は UE4 標準の標準デザイン方法としてオンラインラーニングでも紹介されていますので、
そちらも合わせてご覧ください。

「アンビエントおよびプロシージャル サウンド デザイン」
https://www.unrealengine.com/ja/onlinelearning-courses/ambient-and-procedural-sound-design

パラメータコントロールベースデザイン

サウンドのパラメータコントロールベースデザインは、
数値を適用するとその数値に沿うように鳴らし方を変えていく手法のことを言います。

例えば、次のようなケースがパラメータコントロールベースデザインになります。

■デザイン例

・スポーツの試合における盛り上がり度に応じた歓声の大きさ
　盛り上がり度をパラメータ化する。(冷めている 0 ～ 盛り上がっている 1)
　自チームが負けているとパラメータは 0 に近づいていき落胆の声が強くなってきて、
　相手チームの歓声が盛り上げっていく。
　自チームが勝っているとパラメータは 1 に近づいていき盛り上がりの声が強くなっていき、
　相手チームの歓声が落胆の声になっていく。

・天気に応じた雨音の大きさ
　天気をパラメータ化する。(晴天 0 ~ 豪雨 1)
　晴天時は雨音はなく、天気が悪くなるにつれパラメータが 1 に近づいていき雨音が強くなってくる。

・ゲームの進行度に応じた BGM の遷移
　ゲームの進行度をパラメータ化する。(スタート 0 ～ ゴール 1)
　BGM をパートごとに区切りパート単位でループできるようにする。
　ゲームが進行してパラメータが 1 に近づいていくと、BGM のパートが遷移していく。

■実装例

天気に応じたパラメータ変化について具体的に見ていきます。

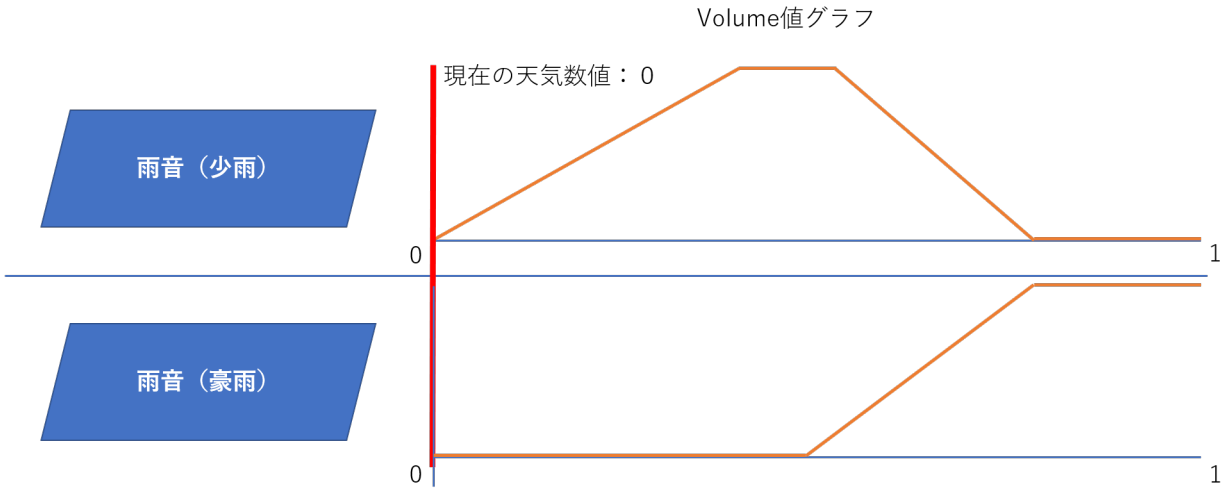
天気の度合いを晴天 0 ~ 豪雨 1 としたときに、雨音をこのパラメータに応じて、
少雨→豪雨という感じに連続的に音の変化を与えていきたいとします。

よくある方法としては、少雨の音、豪雨の音を別々に用意して、
天気の度合いに応じてこの二つの音のボリュームを変化させていく、というものがあります。
晴天の際には雨が降っていないので、少雨の音、豪雨の音がともにボリュームが 0 となりますし、
天気が少雨に近付く（天気が悪くなってくる）につれて少雨のボリュームが 1 に近づいていきます。
雨が降り出した後、天気が少雨と豪雨の間くらいの場合には少雨のボリュームが 0.7 くらい、
豪雨のボリュームが 0.4 くらいと徐々に少雨のボリュームが小さくなっていき、
逆に豪雨のボリュームが大きくなっていくというようにボリューム値を変化させていきます。

もちろん、別々に制御するということは可能ですが、昨今のミドルウェアではこうした二音を一つにまとめて
パラメータを変化させていけば、ボリューム値などをそれに応じて変化させていくという仕組みがあります。
ＣＲＩではその機能を「AISAC」と呼ぶ機能で行っていきます。

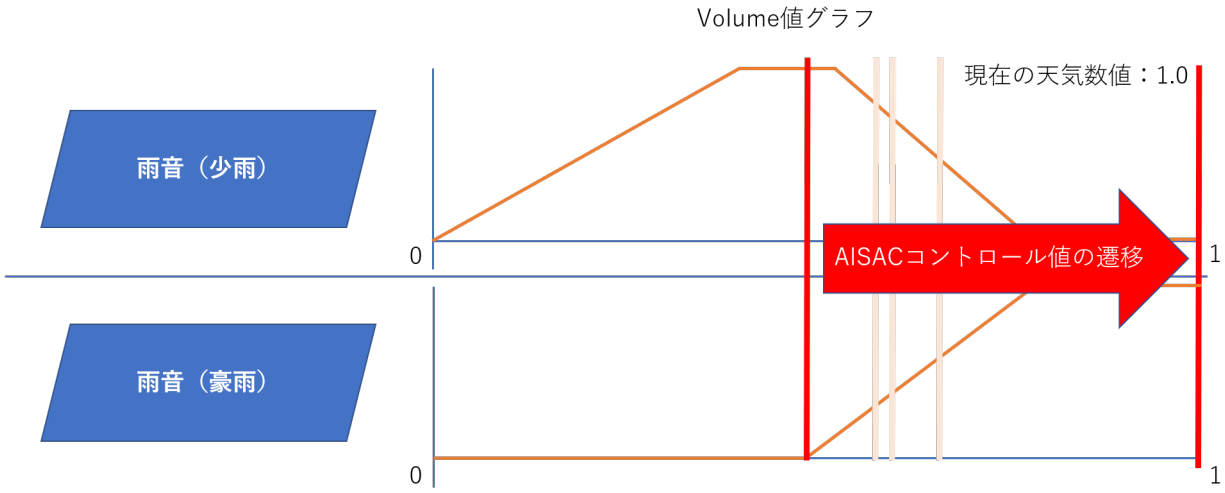
ここに少雨と豪雨のボリューム値をカーブであらわした図があります。
縦軸はボリューム値、横軸は天気の度合い（AISAC コントロール値）になります。

天気：0（晴れ）～1（大雨）



AISAC コントロール値が 0 の場合、つまり晴天の場合には雨音がないため、少雨と豪雨の音量は 0 になります。
少雨から豪雨に変わっていく際には AISAC コントロール値は 0.5 ～ 1.0 に遷移していき、
それに合わせて少雨と豪雨のボリュームが入れ替わるように変化していきます。

天気：0（晴れ）～1（大雨）



このときに、AISAC コントロール値を変化させるだけでよいということがポイントになります。
二つのサウンドのボリュームを一つのパラメータで変化するようまとめることで、
個別に音のボリュームを制御するという手間を省いていくことができます。

このように、複数のサウンドのパラメータを一つにまとめて制御できるようにするということが
パラメータコントロールベースデザインになります。

chapter 3 アンビエントサウンドの実装（仕様策定）

この章では、前章で取り上げたプロシージャルサウンドデザイン、
パラメータコントロールベースデザインの二つのデザイン方法を用いて、
ADX2 でアンビエントサウンドを実装するにはどうしたら良いのか、
「鳥の鳴き声」と「雨音」という二種類の事例に沿って説明していきます。

アンビエントサウンドの仕様策定

前章では、アンビエントサウンドを制作するにあたっては以下の点が重要と述べてきました。

・音の再生に規則性がなく不連続
・停止するまで再生が続く
・データ量小さい
・ゲームの状況に応じてサウンドのなり方が動的に変化する

これらを満たすようなアンビエントサウンドをデザインしていきます。

サウンドを付与する環境

実装していくにあたり、下記画像のようなレベルにアンビエントサウンドを追加していきます。

レベルギミック

・雨が振ったり止んだりする

※レベルに配置してあるアセットはマーケットプレイスから以下の名前のものとなっています。

・町のモデル：「Modern City Downtown with Interiors Megapack (Modular Urban Buildings)」
・雨等のエフェクト：「Soul: City」

アンビエントサウンドの演出仕様

今回は次の二つのアンビエントサウンドを作成していきます。

・木々から漏れる鳥の声
・雨音
この二つのサウンドをステージギミックに沿うようデザインしていきます。

■木々から漏れる鳥の声
どのように再生する？
・無限長で再生は停止するまでずっと再生し続ける
・鳴き声というものはコミュニケーション
　短い間隔で二回もしくは三回ランダムに鳴き声を再生することでコミュニケーションを表現する
・天気の切り替わり（晴れ、雨、土砂降り）で鳴き声の頻度が変わる
　晴れ（たくさん）→雨（少し静かになる）→土砂降り（ほぼなし）

■雨音
どのように再生する？
・無限長で再生は停止するまでずっと再生し続ける
・雨の強さをパラメータによって連続的に変更できる。
・天井がある場所にプレイヤーが入ったら、高周波成分をカットして雨がさえぎられていることを表現する。

各アンビエントサウンドの実装仕様

■木々から漏れる鳥の声
どのように再生する？
・すべての木の上部で鳥の鳴き声が聞こえるようにする
・レベル上に配置したらずっと再生し続ける
・開始時刻と再生タイミングランダムの調整でランダムな再生タイミング制御を行う
・開始時刻と再生タイミングランダムを別に設定したトラックを複数用意し、ゲーム変数を使って雨音の
　強さ（AisacControl 値、RainPower）に応じた段階的な再生頻度変化を与える

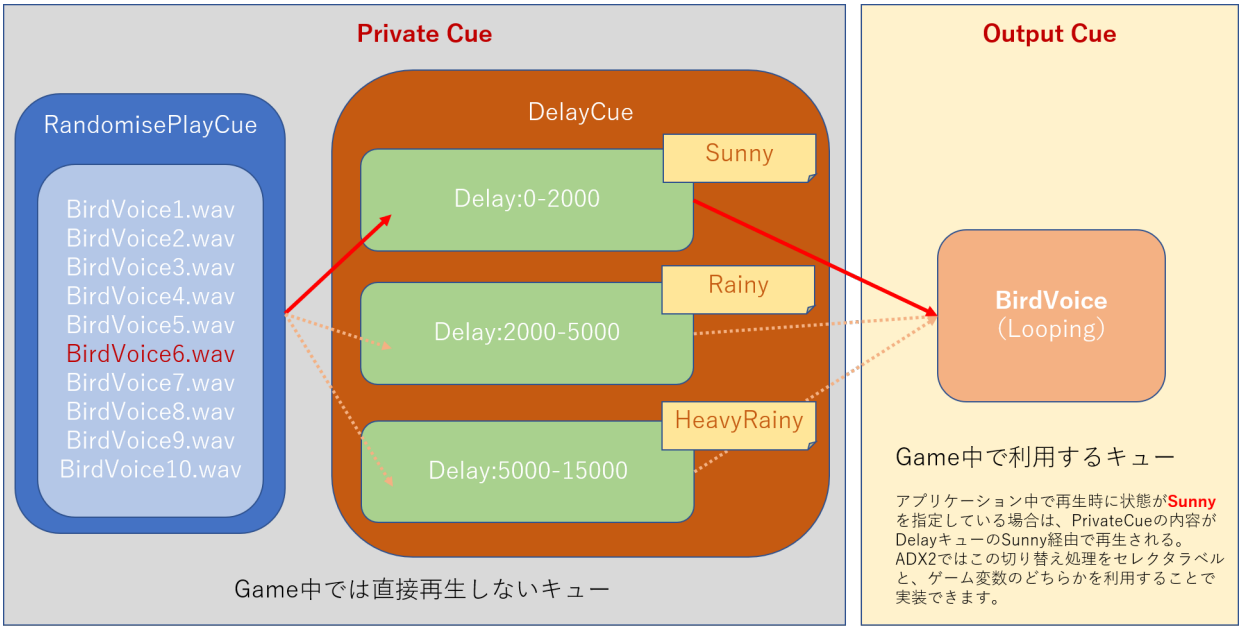
■雨音
どのように再生する？
・木のアクター BP に AtomComponent を配置して常に再生できるようにする。
・レベル上に配置したらずっと再生し続ける。
・雨の強さを AisacControl 値（RainPower）によって連続的に変更できるようにする。
・天井がある場所にプレイヤーが入ったら、高周波成分をカットして雨がさえぎられるよう、
　別の AisacControl 値（BarriorRate）で連続的に変更できるようにする。

chapter 4 アンビエントサウンドの実装（鳥の鳴き声編）

さて、「鳥の鳴き声」を実際にミドルウェアで設定してみましょう。

木々から漏れる鳥の声のミドルウェア設定

木々から漏れる鳥の声は先述したプロシージャルデザインで制作していきます。
実装イメージとしてはこちらの図のようなイメージです。



BirdVoice キューでの再生を行うことで、RandomisePlayCue でランダムに波形選択が行われ、
その RandomisePlayCue が再生されるタイミングを DelayCue でタイミング遅延を行うというように、
機能をキュー単位で細かく区切って実装していきます。
今回はこうしたノードベースで組めるようなフローを ADX2 ではどのように組んでいけるのかということについても示していきます。

※今回は一つのモジュールのみですが、BirdVoice に別のフローを追加して複数のモジュールを同時に制御するというようなキューも作っていくことができます。

素材の用意

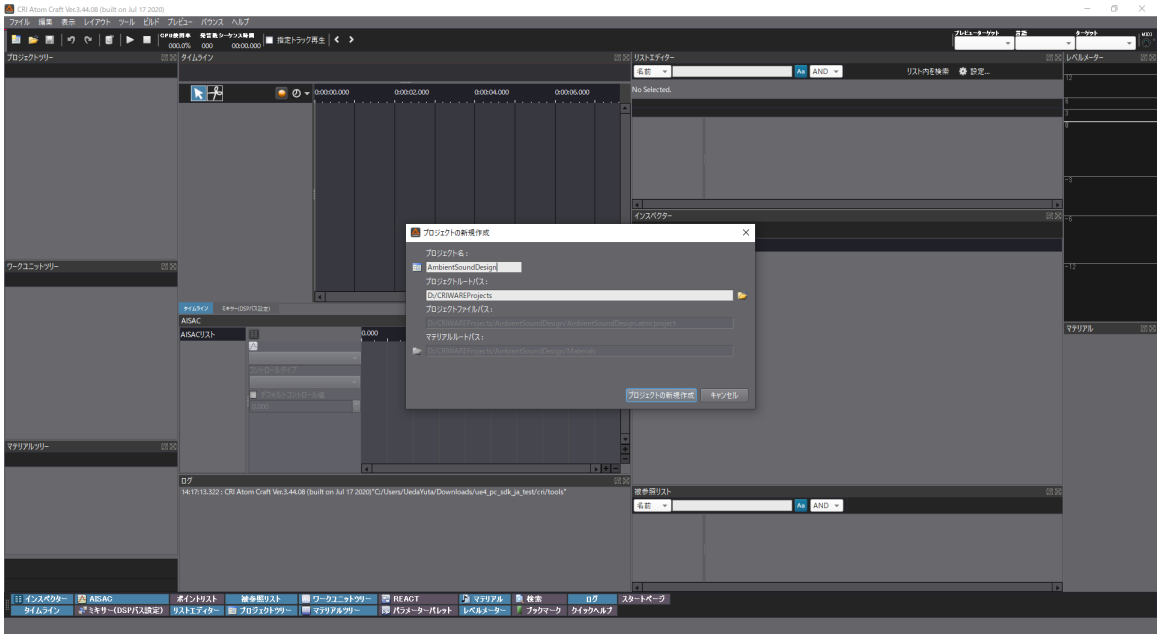
今回は鳥の鳴き声を表現するため、鳴き声を細かい単位で収録した wav ファイルを用意します。

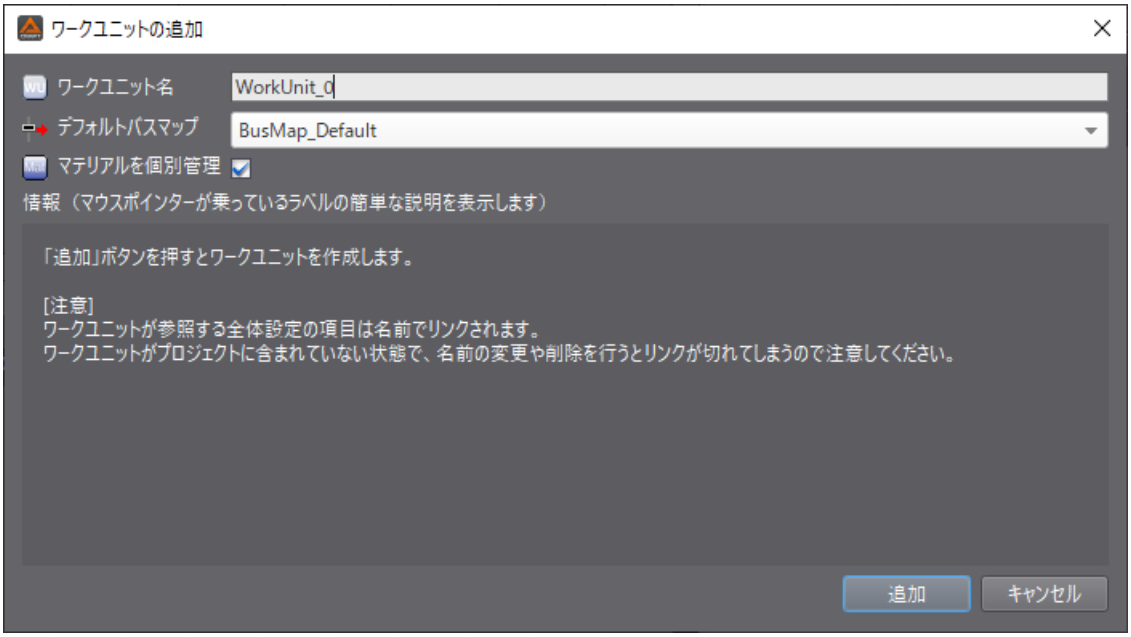
CRI Atom Craft での実装

wavデータをADX2のサウンドオーサリングツール「CRI Atom Craft」にインポートして実装を行っていきましょう。

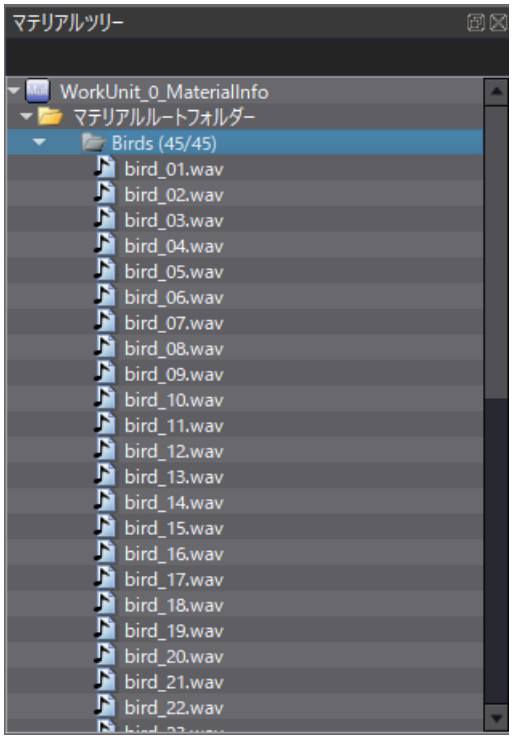
[BirdVoice キューの作成]

1．CRI Atom Craft を立ち上げて新規プロジェクトを AmbientSoundDesign という名前で作成します。

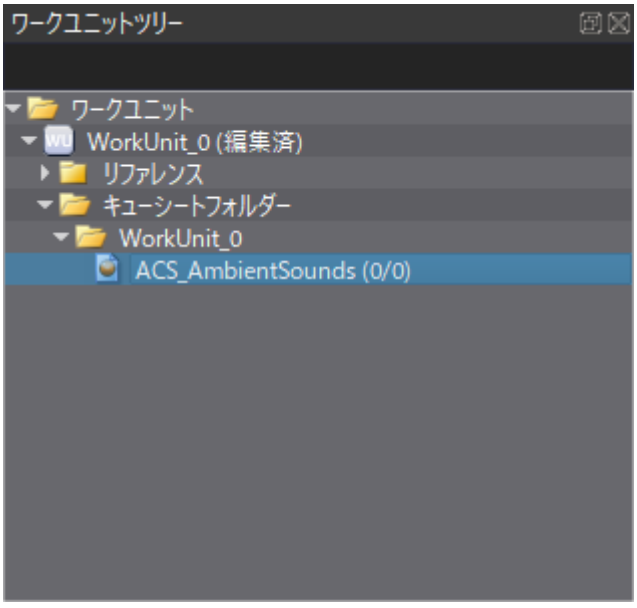




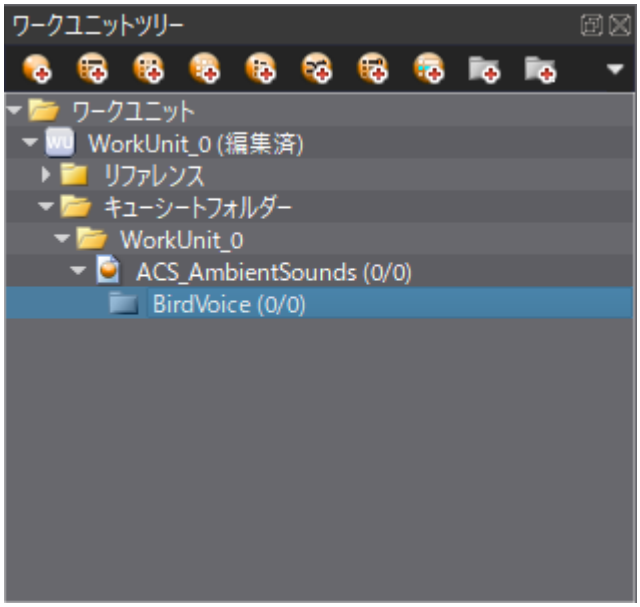
2．用意した素材を CRI Atom Craft にインポートします。



3．CRI Atom Craft のワークスペース内に「ACS_AmbientSounds」という名前の CueSheet を作成します。

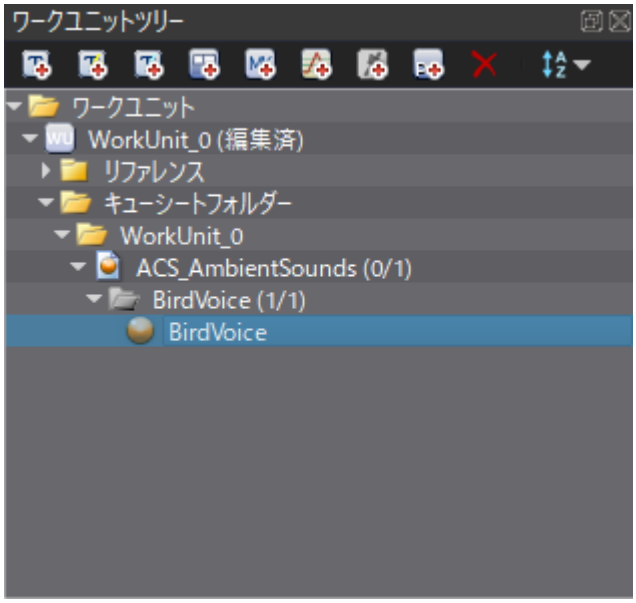


４．ACS_AmbientSounds CueSheet 内に BirdVoice という名前のキューフォルダーを作成します。

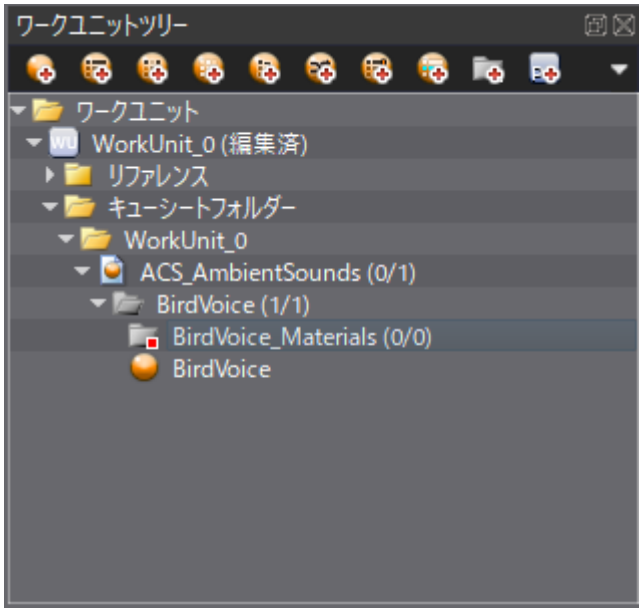


キューフォルダーを作成することで、そのフォルダ内部でキューを作成し処理を作っていくことができるため、
階層管理がより分かりやすくなります。
今回は、プロシージャルデザインで複数のキューを組み合わせた再生方法を作っていくことになるため
関連のあるキューを一つのフォルダにまとめていきます。

５．BirdVoice キューフォルダー内に BirdVoice という名前のキューを作成します。

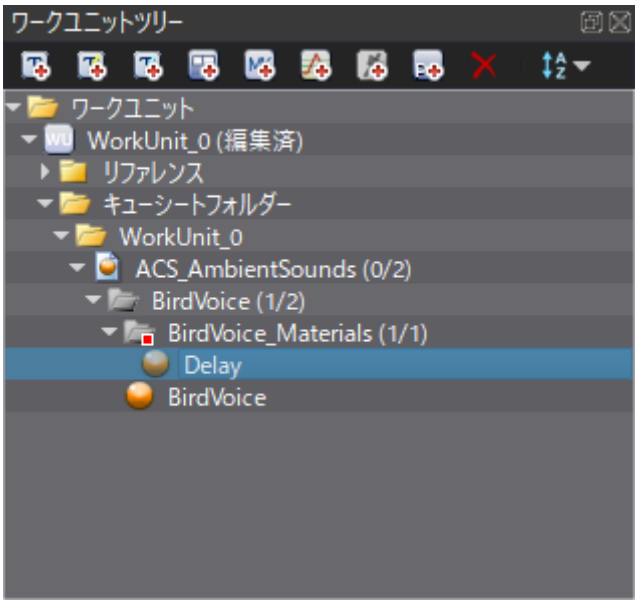


6．BirdVoice キューフォルダー内に BirdVoice_Materials という名前の
プライベートキューフォルダーを作成します。

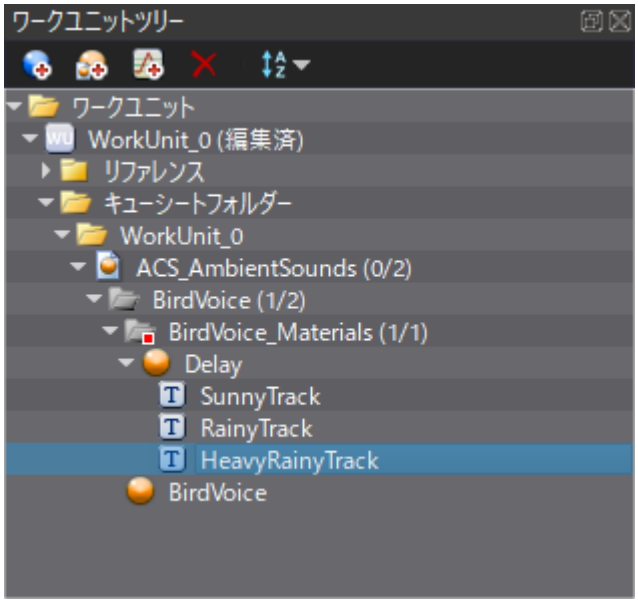


※プライベートキューフォルダーとは？
通常、作成したキューは、データをビルドした際に全てゲーム中から直接呼び出すことができます。
しかし、その中には別のキューを用いて再生するなど、ゲーム中では直接呼び出さないキューも存在し、
そういった場合にはアクセスできないようにすることが望ましいです。
ゲーム中の直接的なアクセスを制限するための機能としてあるのが、プライベートキューフォルダーになります。
プライベートキューフォルダー内部に追加したキューはゲーム中にアクセスできなくすることができるため、
UE4 では不要なキューアセットを作成しないようにもできるため有用です。

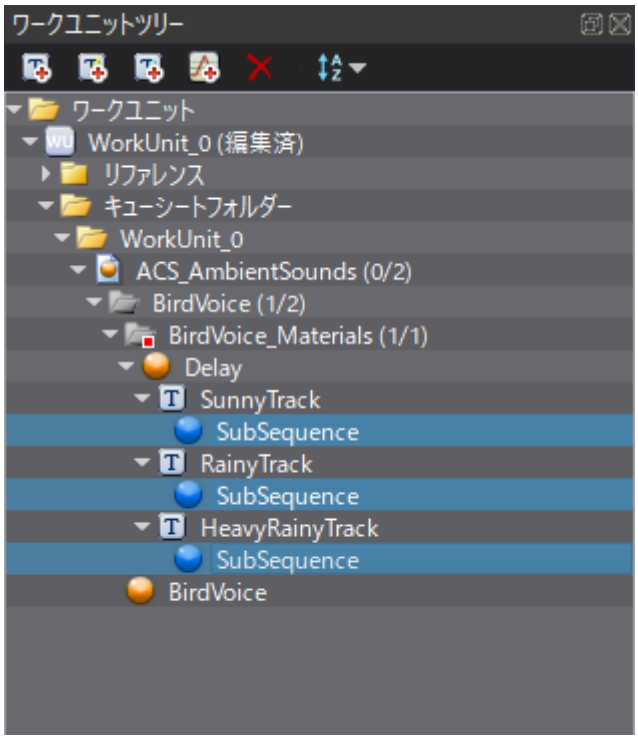
7．BirdVoice_Materials フォルダ内に Delay キューを作成します。
この Delay キューでは、再生時にトラックに配置されたサウンドデータに対して
ランダムに再生処理の発行を遅らせる処理を追加していきます。



8．状況に応じて再生処理発行の遅延時間を変えられるようにするため、Delay キュー内に 3 つのトラックを作成し、
それぞれ SunnyTrack、RainyTrack、HeavyRainyTrack と名前を変えます。

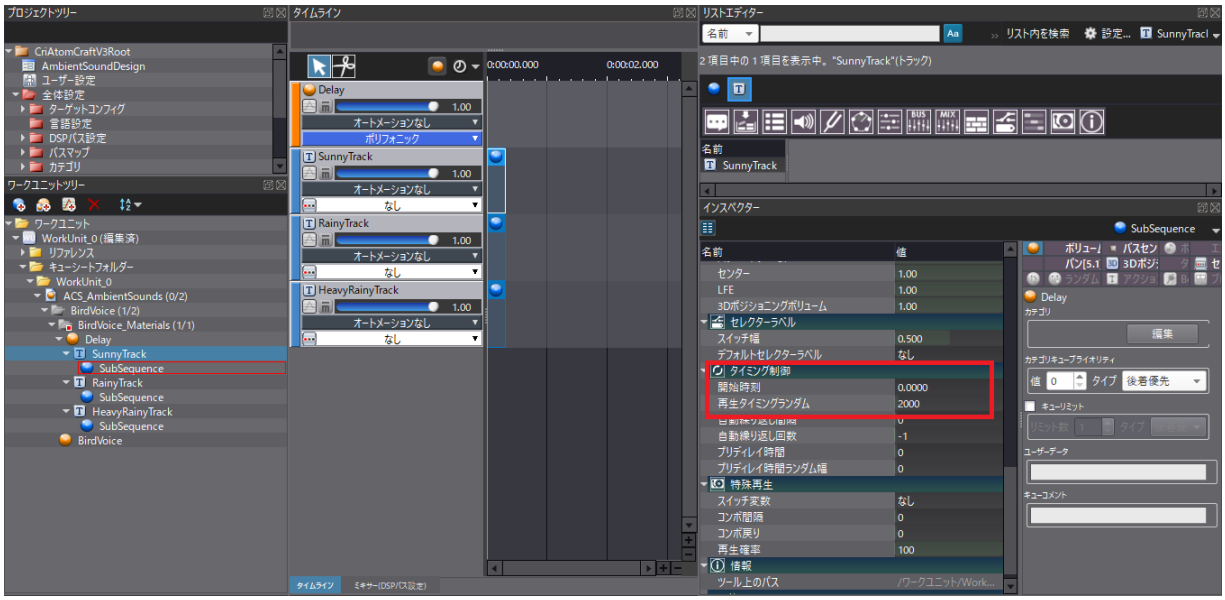


9．Delay キュー内の 3 つのトラックにそれぞれ SubSequence を作成します。



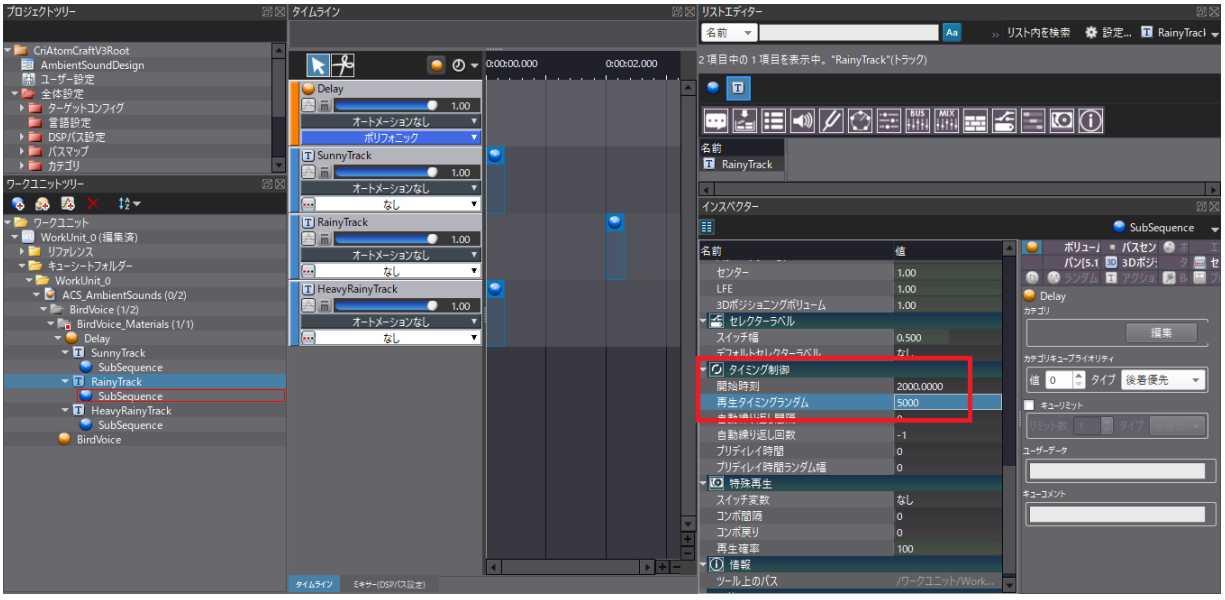
※ SubSequence とは？
サブシーケンスとは、トラックの下に作成可能な、キューと同等の機能を持つオブジェクトです。
キューと同様に、サブシーケンス内にトラックやウェーブフォームリージョンを追加、編集することができます。
ツリー上のサブシーケンスを選択すると、キューのタイムライン表示と同様にサブシーケンスが持つ
タイムラインが表示されます。

１０．SunnyTrack の SubSequence を選択し、
　　詳細パネル中の開始時刻を 0、再生タイミングランダムを 2000 で指定します。

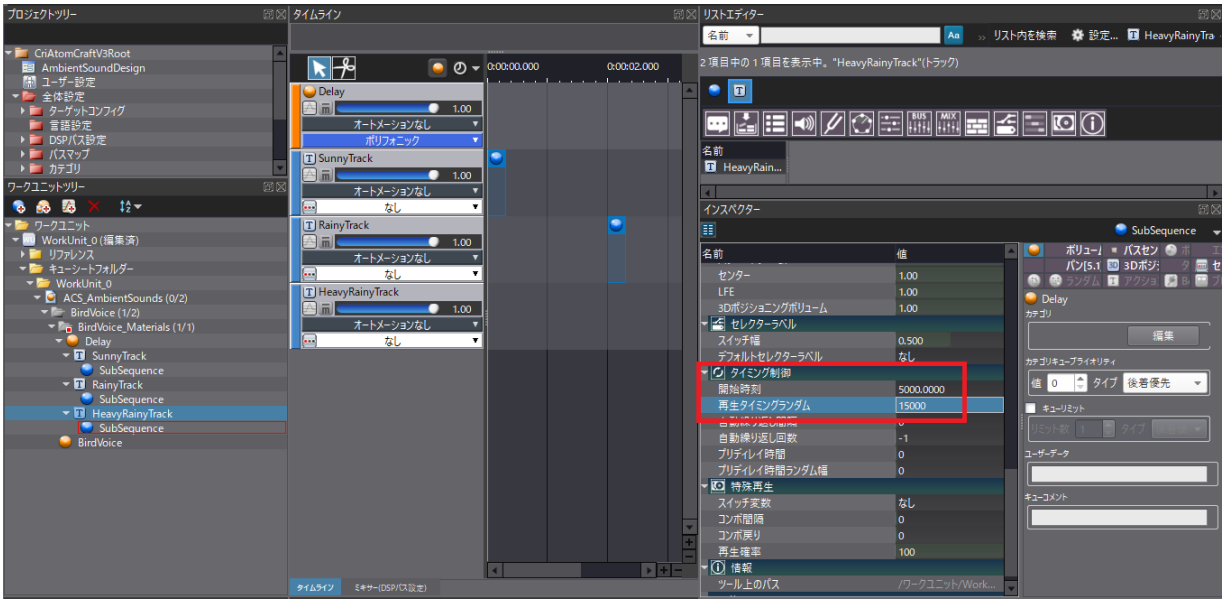


※再生タイミングランダムとは？
再生タイミングランダムとは、開始時刻に対して再生タイミングランダムで設定した数値内で、
さらにランダムに再生タイミングをずらす機能になります。
開始時刻と再生タイミングランダムを組み合わせることで、
サウンドの再生タイミングのランダム範囲遅延を行っていくことができます。

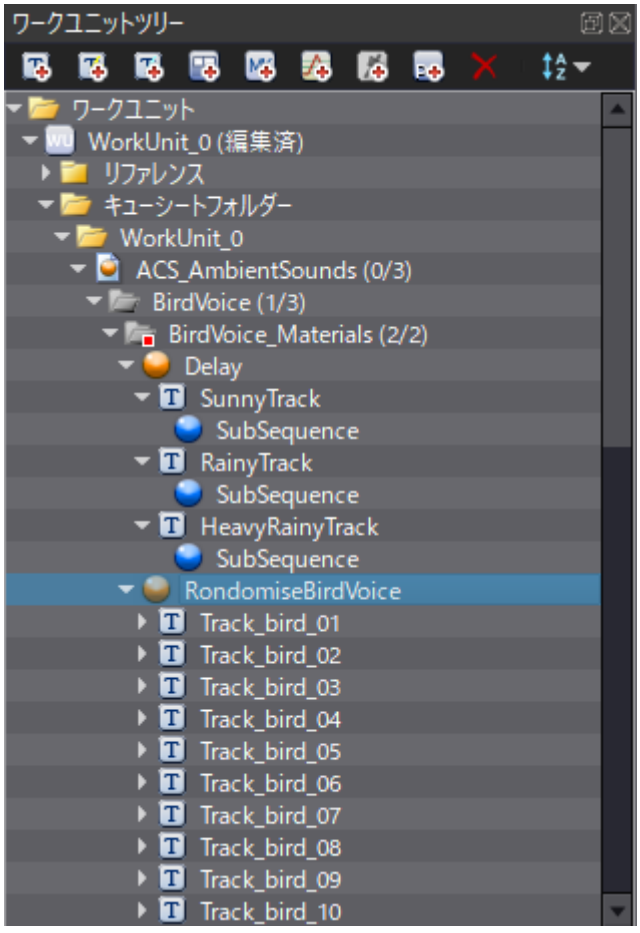
１１．RainyTrack の SubSequence を選択し、
　　詳細パネル中の開始時刻を 2000、再生タイミングランダムを 5000 で指定します。



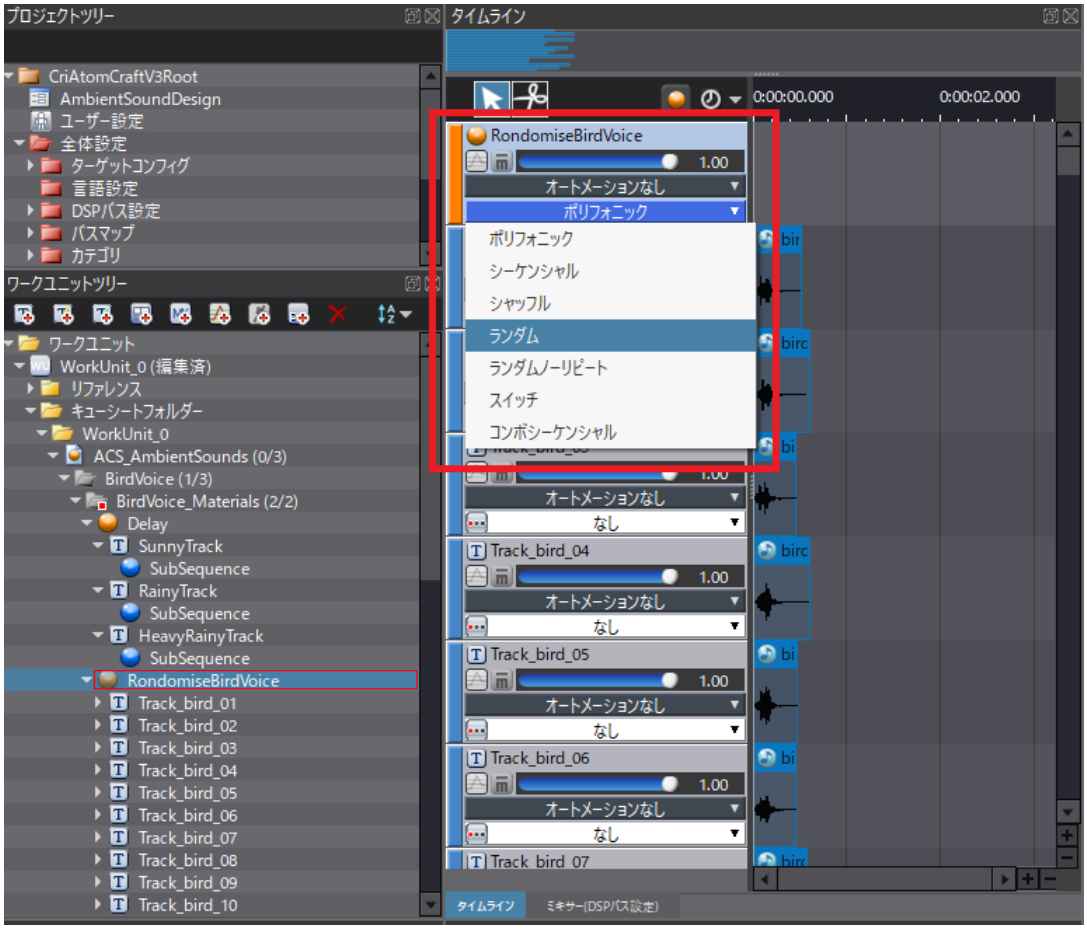
１２．HeavyRainyTrack の SubSequence を選択し、
詳細パネル中の開始時刻を 5000、再生タイミングランダムを 15000 で指定します。



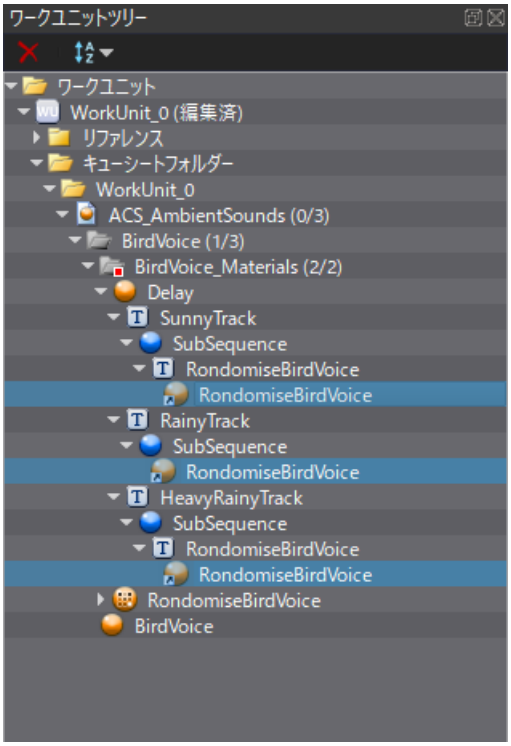
１３．BirdVoice_Materials フォルダ内に RondomiseBirdVoice という名前のキューを作成します。
Import した Bird_*.wav を全部選択して、ワークユニットの RondomiseBirdVoice 上に
ドラッグ & ドロップします。



１４．RondomiseBirdVoice キューを選択し、キューのシーケンスタイプをランダムに変更します。

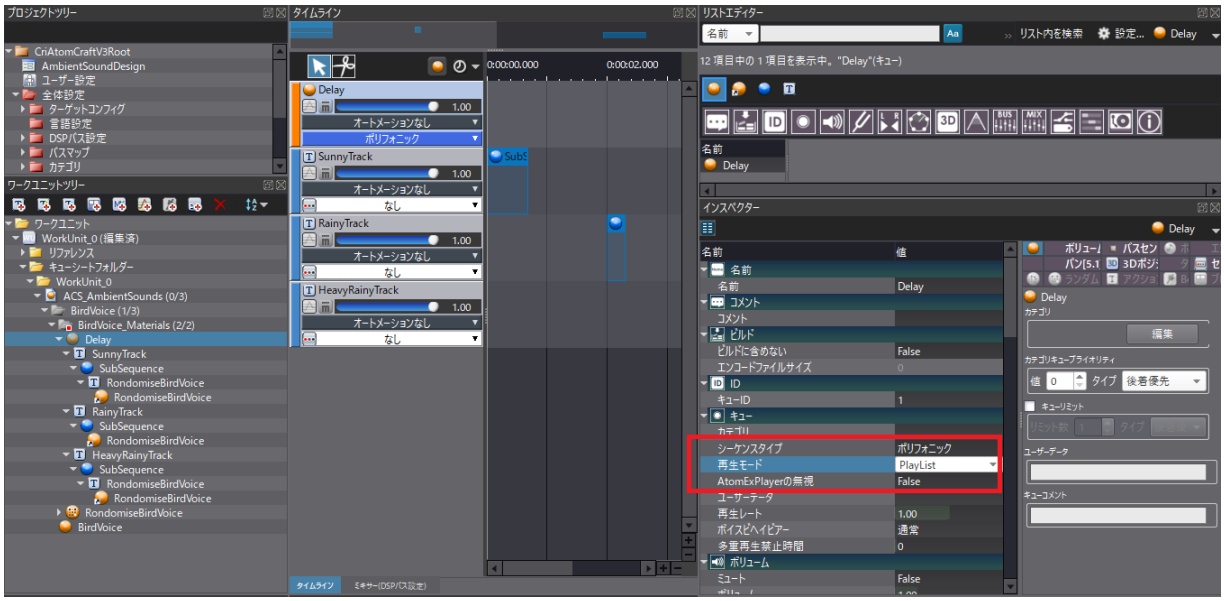


１５．RondomiseBirdVoice キューを選択し、Delay キュー内の SunnyTrack、RainyTrack、HeavyRainyTrack の SubSequence 内にドラッグアンドドロップします。



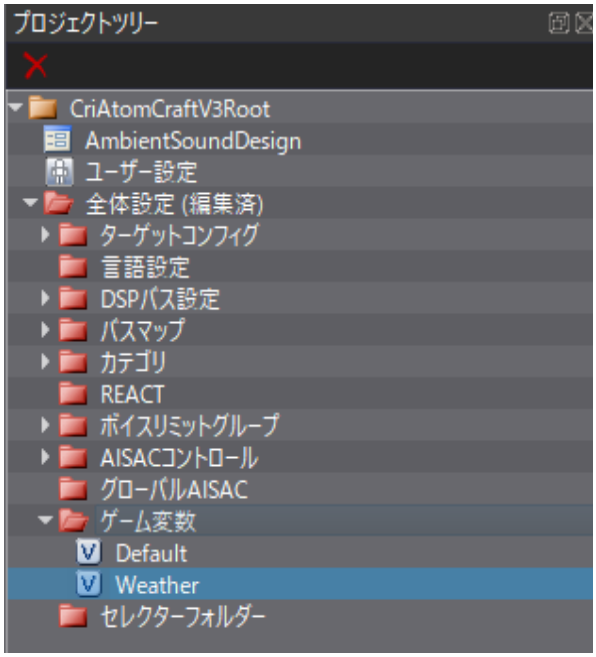
※キューリンク
キューを別のキュー内に配置して参照状態にすることをキューリンクといいます。
この機能を利用することで、一つの親キューで複数の子キューを制御できるようになり、
プログラマブルな制御がやりやすくなります。

１６．Delay キューを選択して、再生モードを Playlist に変更します。
Playlist のモードは、事前に設定してあるキューのシーケンスタイプによって動作は変わります。
今回はランダムで設定しているため、Playlist モードでの再生はキュー単位でのループ再生となり、
キューの内容を繰り返すようになります。

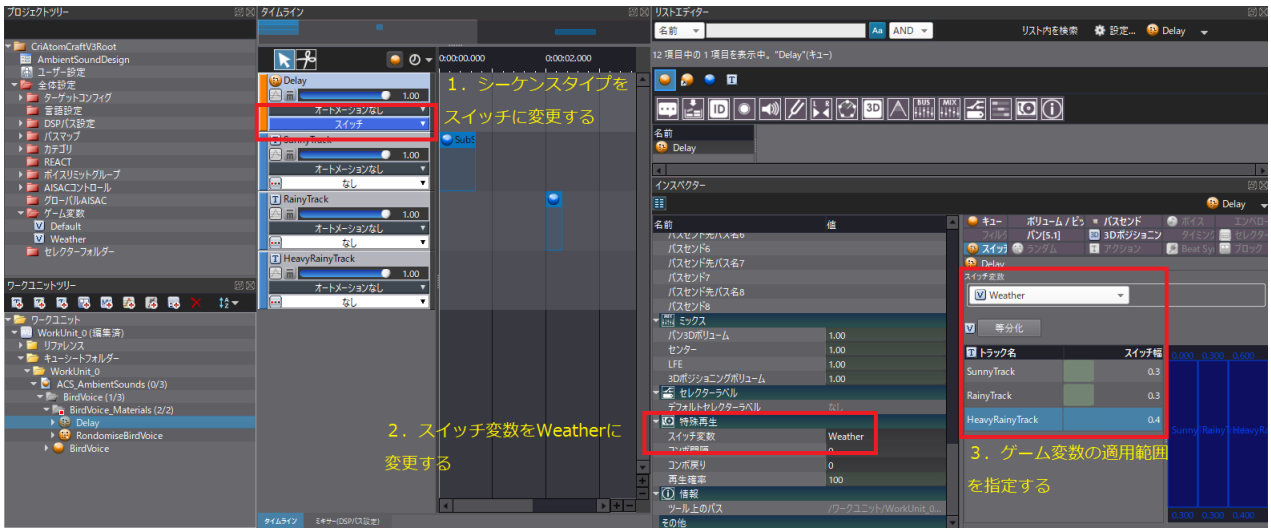


※ Playlist モードでの再生について
基本的にはキューの再生モードを Playlist にするとキューの再生が繰り返されるものになります。
しかし、シーケンスモードがシーケンシャルとシャッフルの場合は複数トラックがキューに配置されている場合、
順番通り、順序ランダムはあれど、順番にトラックが再生され、
すべて再生されると最初からトラックを一つずつ再生していく動作になります。

１７．プロジェクトツリーのゲーム変数フォルダを右クリックし、
Weather という名前のゲーム変数を作成します。



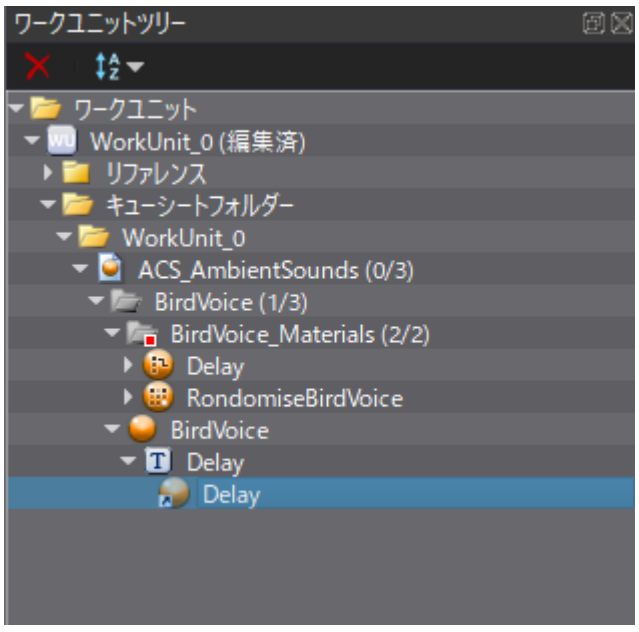
１８．Delay キューの詳細パネルの Switch 変数を作成した Weather ゲーム変数に設定します。
　　加えて、詳細パネル中の Switch カテゴリを選択し、スイッチ幅を次のように設定します。



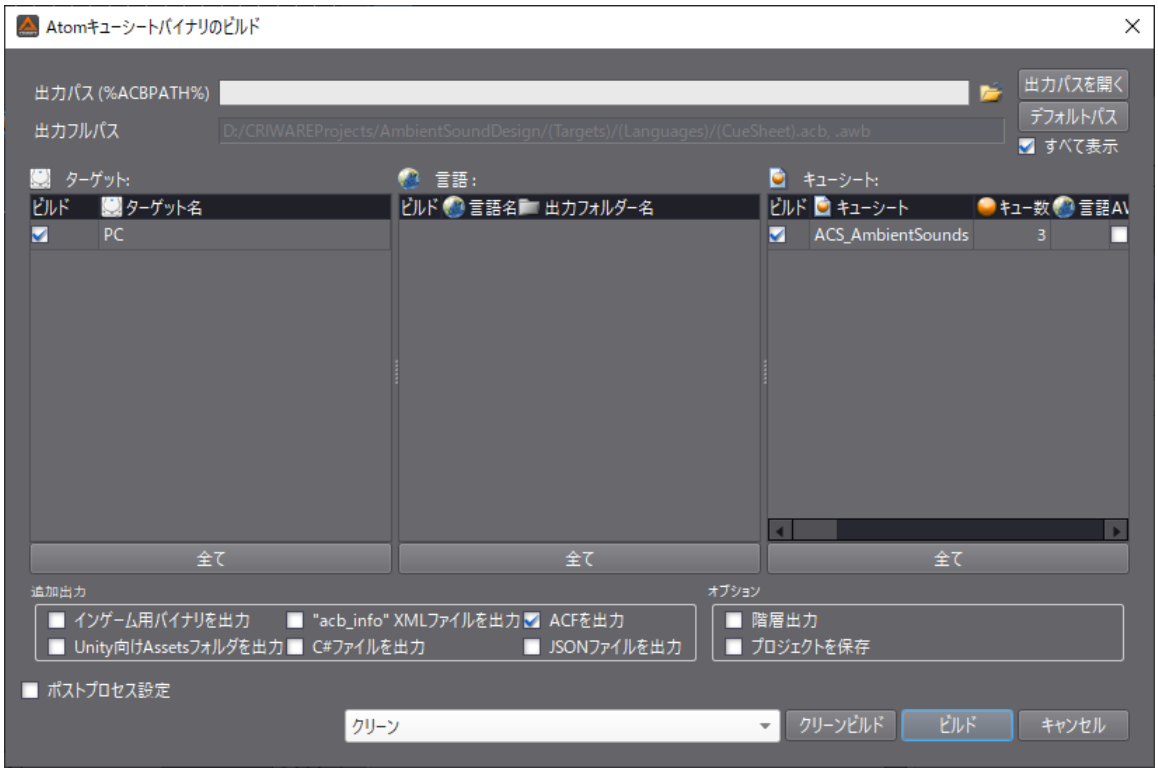
・SunnyTrack:0.3
・RainyTrack:0.3
・HeavyRainyTrack:0.4

この値の指定の意味は、Weather ゲーム変数が 0.0 ～ 0.3 までの範囲は Sunny の状態を示し、
0.3～0.6までの範囲をRainy、0.6～1.0までの範囲はHeavyRainyの状態となるようにするという設定になります。
アプリケーション中で Weather ゲーム変数を動的に変化した際に、Weather ゲーム変数が
どの状態の範囲内にあるかでサウンドの切り替えを行っていくことができるようになります。

１９．Delay キューを BirdVoice キュー内にドラッグアンドドロップします。



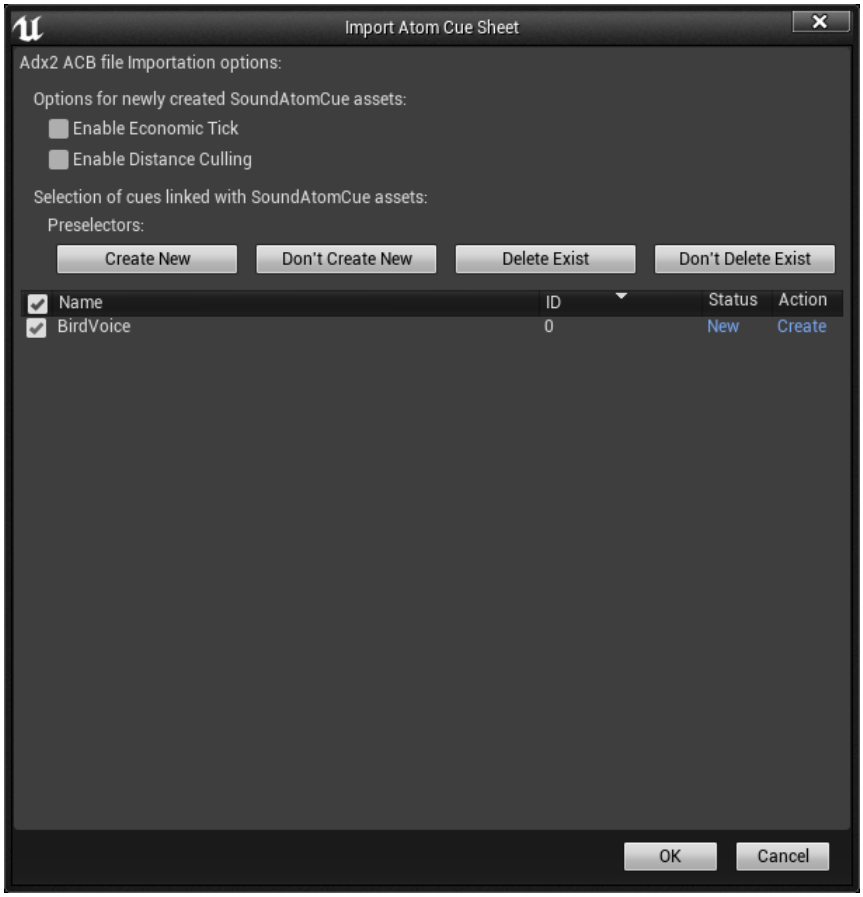
２０．データをビルドします。ビルドすると ACS_AmbientSound.acb と AmbientSound.acf という名前のファイルが出来上がります。この二つのファイルを次項で UE4 にインポートします。



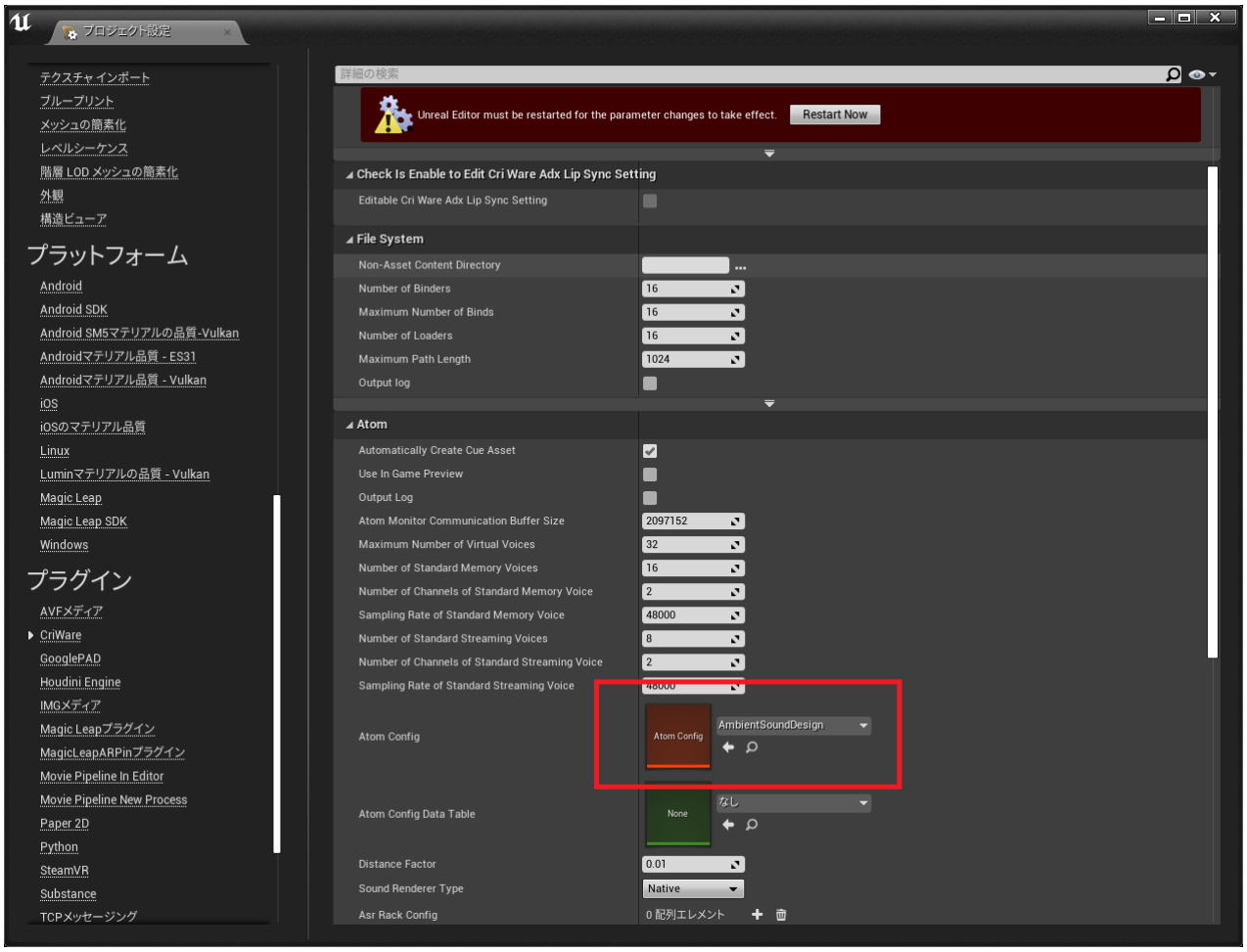
UE4 上での再生

BirdVoice キューをレベル上に配置して、インタラクティブに再生頻度を変えていきます。
BirdVoice キューは、レベル上の木の上部に配置していきます。

1．UE4 に AmbientSound.acb と AmbientSound.acf をインポートします。
　インポートダイアログウィンドウが出ますがそのまま OK を押します。



2．編集タブ→「プロジェクト設定」を選択して、CriWare カテゴリを開き、
AtomConfig 項目に作成された AmbientSound コンフィグアセットをセットして UE4 エディアを再起動します。
パラメータをプロジェクト設定で変更するとウィンドウ上部に Restart ボタンが表示されるため、
ここからリスタートできます。



3．再起動後、BirdVoice アセットを配置していきます。
今回は鳥のモデルの表示などがなく、漠然と木の茂っている場所から鳥の声が聞こえればよいため、
CRI UE4 Plugin の機能の一つ AtomAreaSoundVolume という Volume を使っていきます。

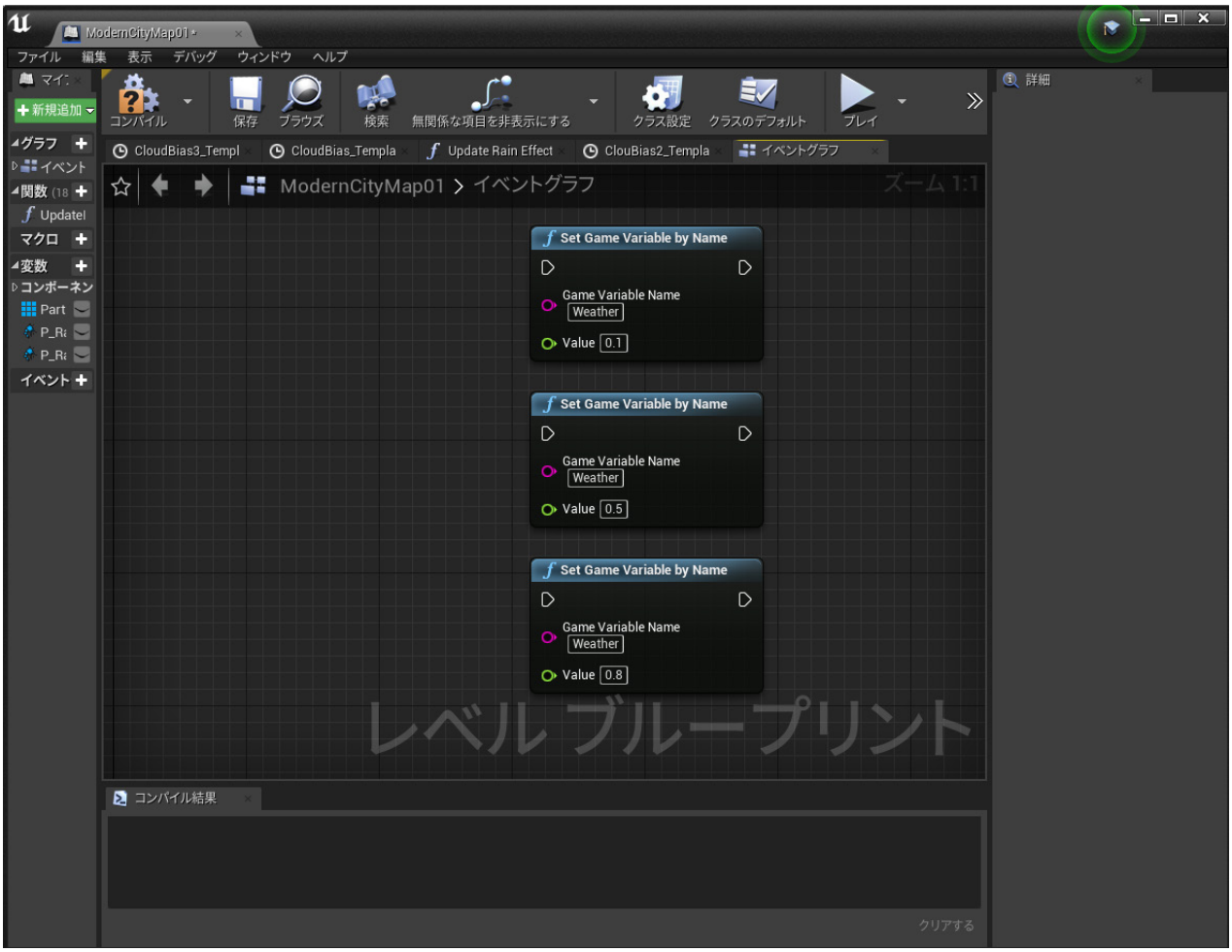
※ AtomAreaSoundVolume とは？
AtomAreaSoundVolume は形状音源を再現するための機能になります。
AtomAreaSoundVolume をレベル上に配置しサウンドを再生すると、サウンドの距離減衰領域はそのボリュームの形状通りとなります。湖や川など比較的広い空間に AreaSoundVolume を配置することで、点音源を多数配置するよりも少ない音数で広い範囲の環境音を配置していくことが可能となります。

実用例：
・川や湖などの広大なフィールドに対する環境音配置

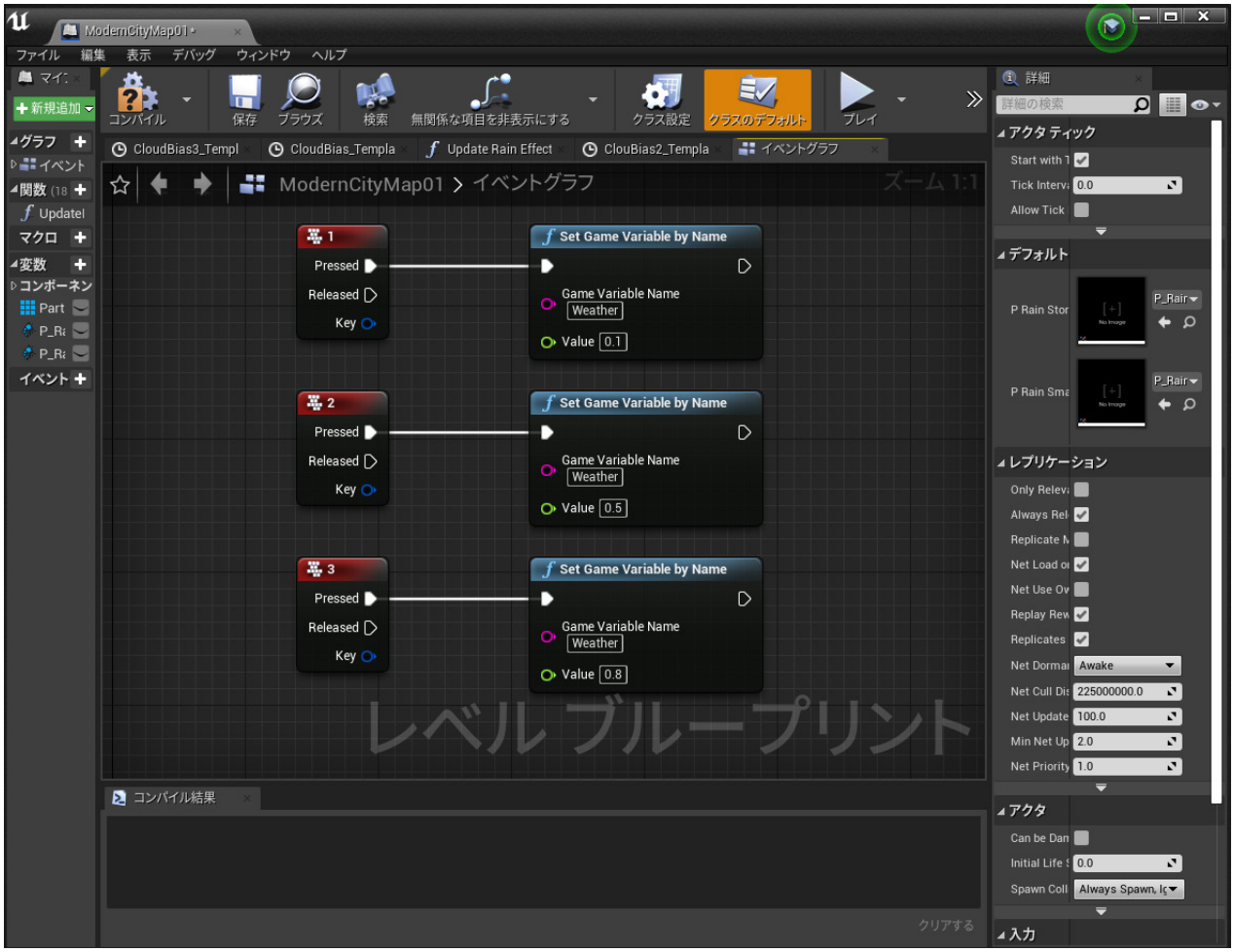
4．木の草が茂っている場所を複数覆うようにして、AreaSoundVolume を配置していきます。
形状を Box のままにして図のように 4 つの Volume を、公園を覆うように配置します。
サイズの設定はブラシセッティングの X、Y、Z のパラメータを変更することで調整してください。

5．AreaSoundVolume をすべて選択して、詳細パネル内の Sounds のエレメントを一つ追加して、
BirdVoice を割り当てます。

6．レベルブループリント上で SetAtomGameVariableByName ノードを 3 つ作成して、
GameVariableName に「Weather」を指定し、それぞれ Value を 0.1、0.5、0.8 で指定します。



7．各 SetAtomGameVariableByName ノードの入力実行ピンにキー入力を割り当てます。



8．PlayInEditor で実行して、キー入力を行うと、それに応じて鳥の鳴き声の再生頻度が変わります。

chapter 5 アンビエントサウンドの実装（雨音編）

この章ではもう一つの実装例として「雨音」をミドルウェアで設定していきます。

雨音はパラメータコントロールベースデザインで制作していきます。

素材の用意

雨脚が弱いときのぱらぱらとした音と、強いときのザーザーとした音を用意します。

・Rain.wav

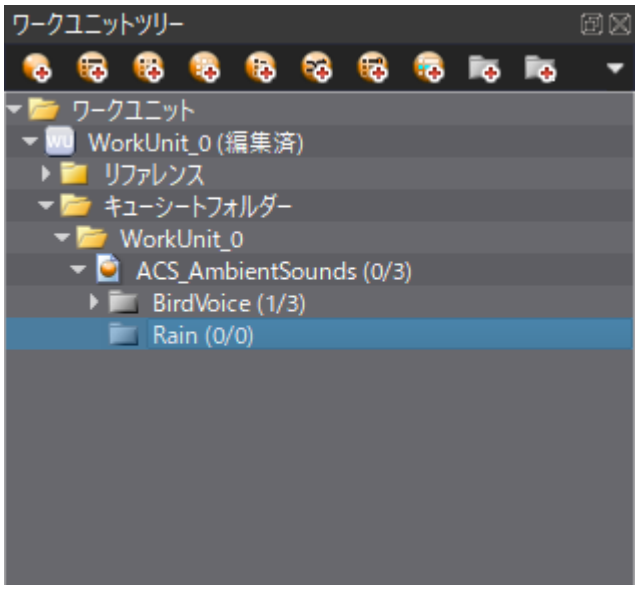
・HeavyRain.wav

CRI Atom Craft での実装

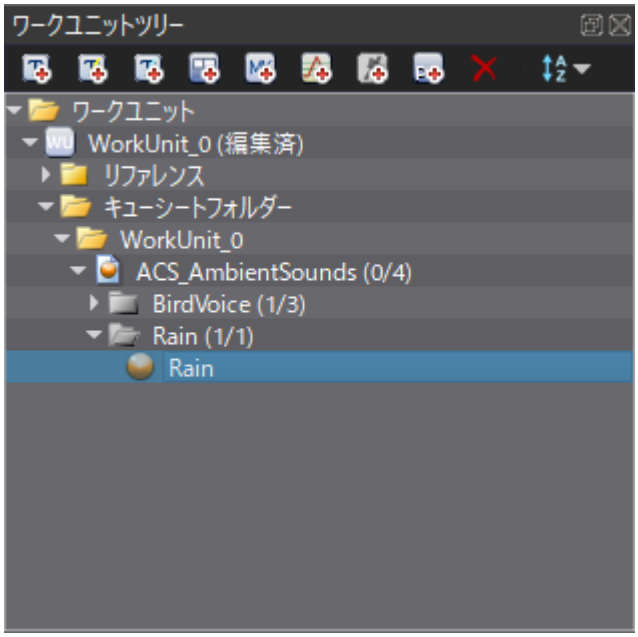
[Rain キューの作成]

1．Rain.wav と HeavyRain.wav を CRI Atom Craft の AmbientSound プロジェクトにインポートします。

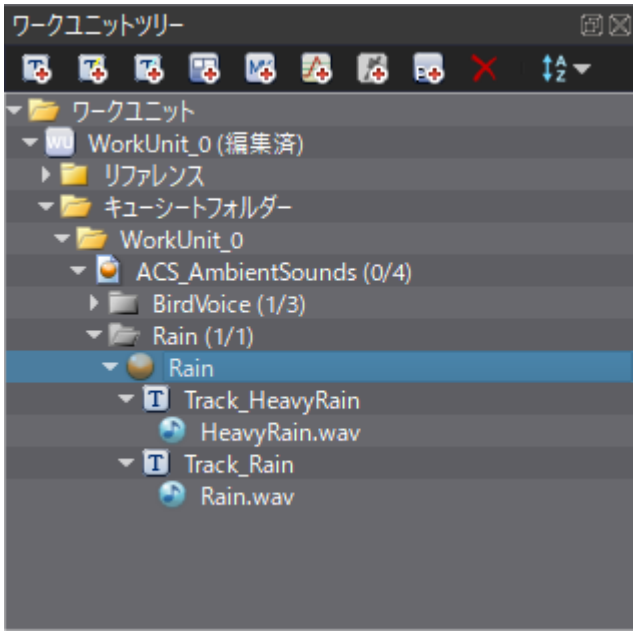
2．ACS_AmbientSounds キューシート内に Rain という名前のキューフォルダーを作成します。



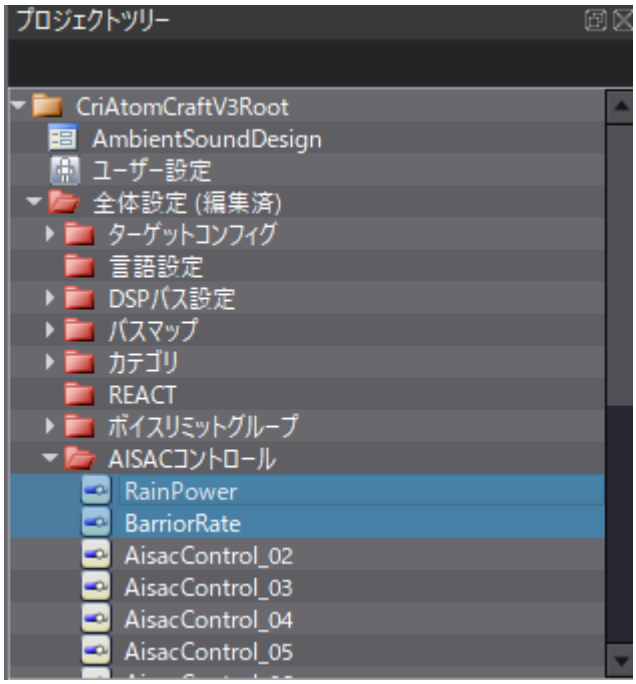
3．Rain キューフォルダー内に Rain という名前のキューを作成します。



4．Rain キュー上に Rain.wav と HeavyRain.wav をドラッグアンドドロップで追加します。

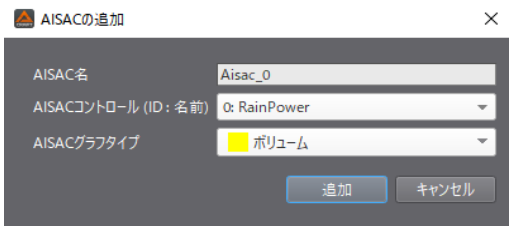


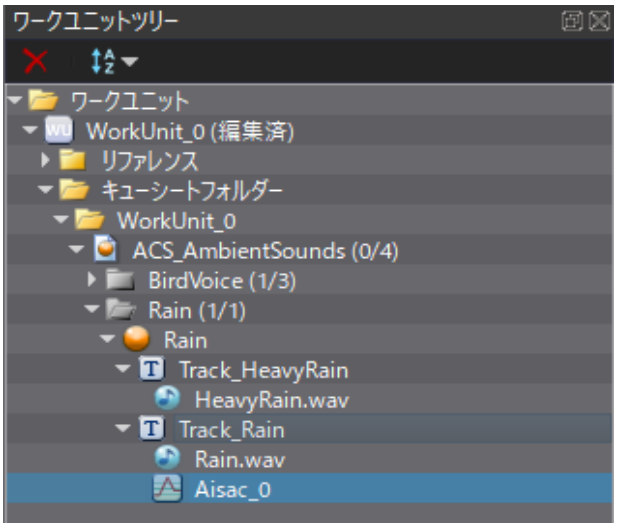
5．CRI Atom Craft のプロジェクトツリーの AISAC コントロールフォルダー内にある、
「AisacControl_00」を「RainPower」、「AisacControl_01」を「BarriorRate」と名前を変更します。



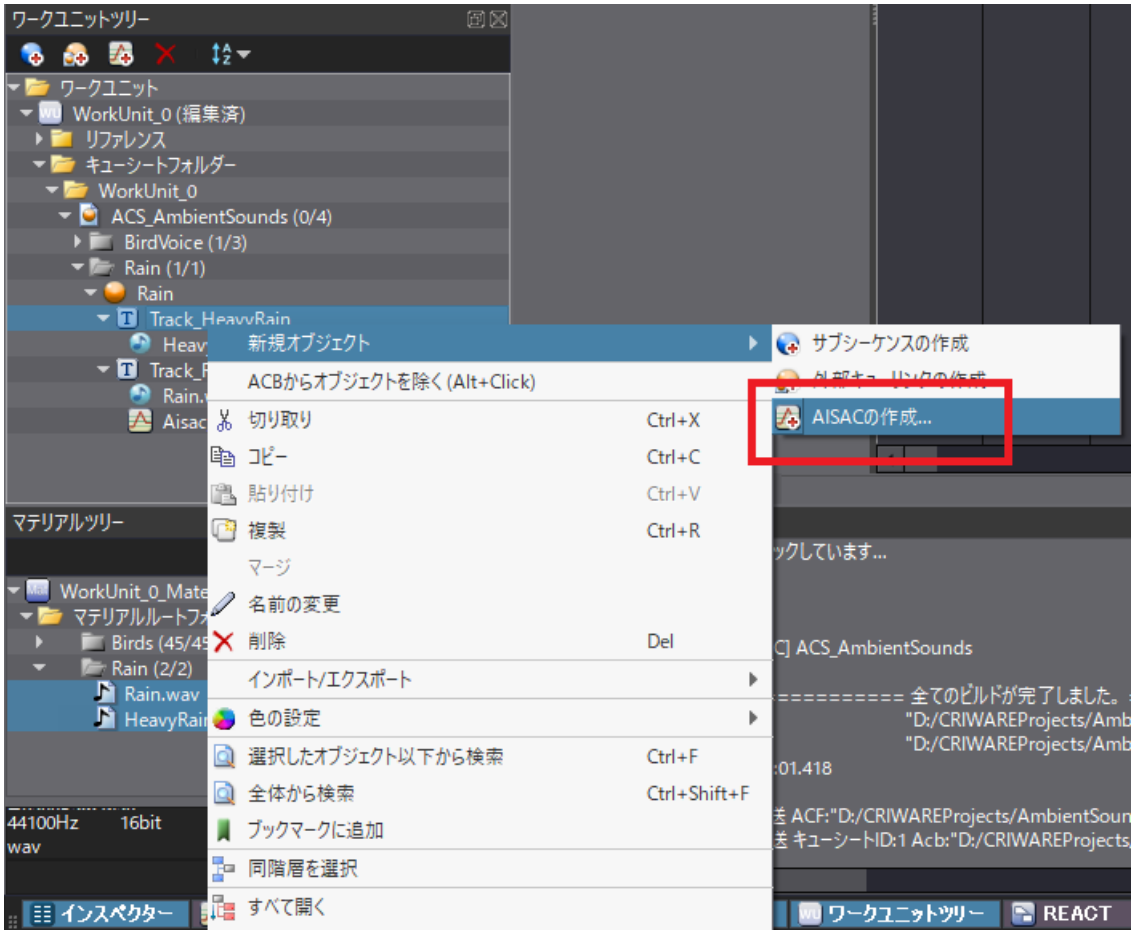
※ Aisac とは？
CRI Atom Craft ツールで再生情報パラメータのカーブを作成し、
そのカーブパラメータをボリュームやバスエフェクト、ピッチなど
様々なサウンドパラメータに適用することができる機能です。
アプリケーション中で Aisac のコントロール用の値を動的に指定することで
カーブのどの場所のパラメータをサウンドに適用するかを決定でき、
ゲームの時間変化やアニメーションの状態変化に合わせて変化することで、
ゲーム世界に合わさったサウンド表現を実現していくことができます。
Aisac はキュー全体に適用するだけでなく、
トラック個別に設定するといったキュー内の部分的なサウンドのみに適用するということができます。

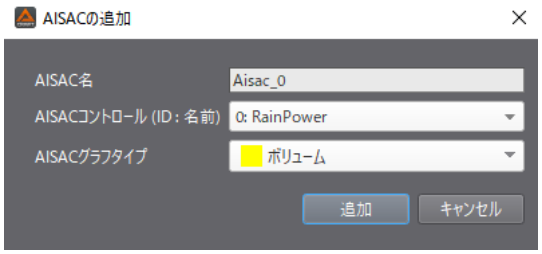
6．Track_Rain を選択し右クリックから Aisac の作成を行い、Aisac コントロールに RainPower と、Aisac グラフタイプにボリュームを指定し追加ボタンを押します。

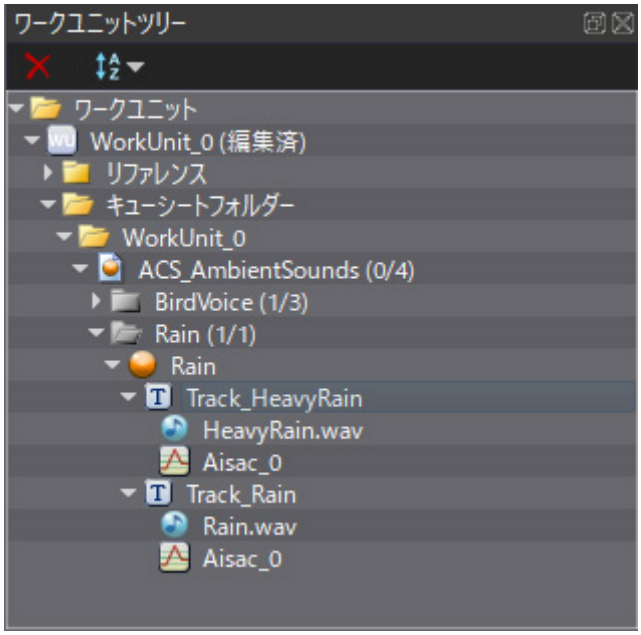




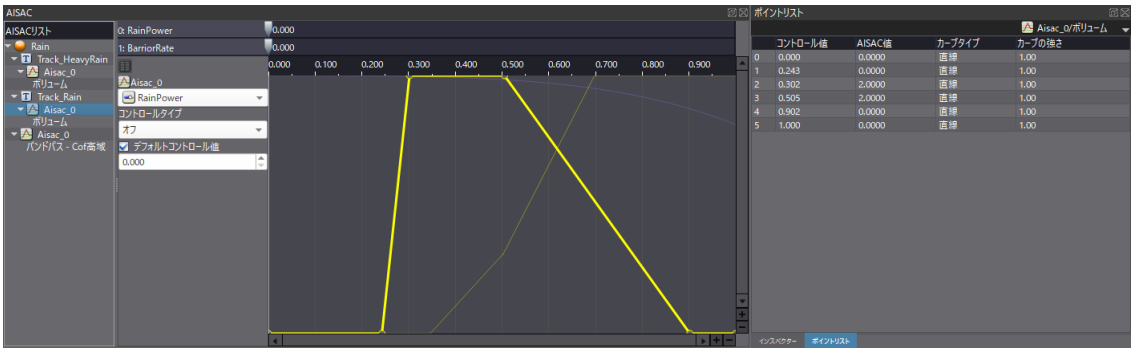
7．Track_HeavyRain を選択し、 6 と同じ操作を行います。



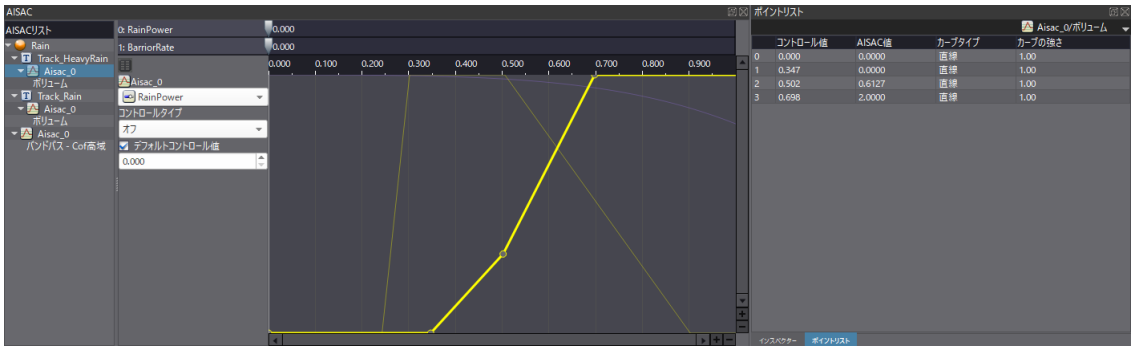




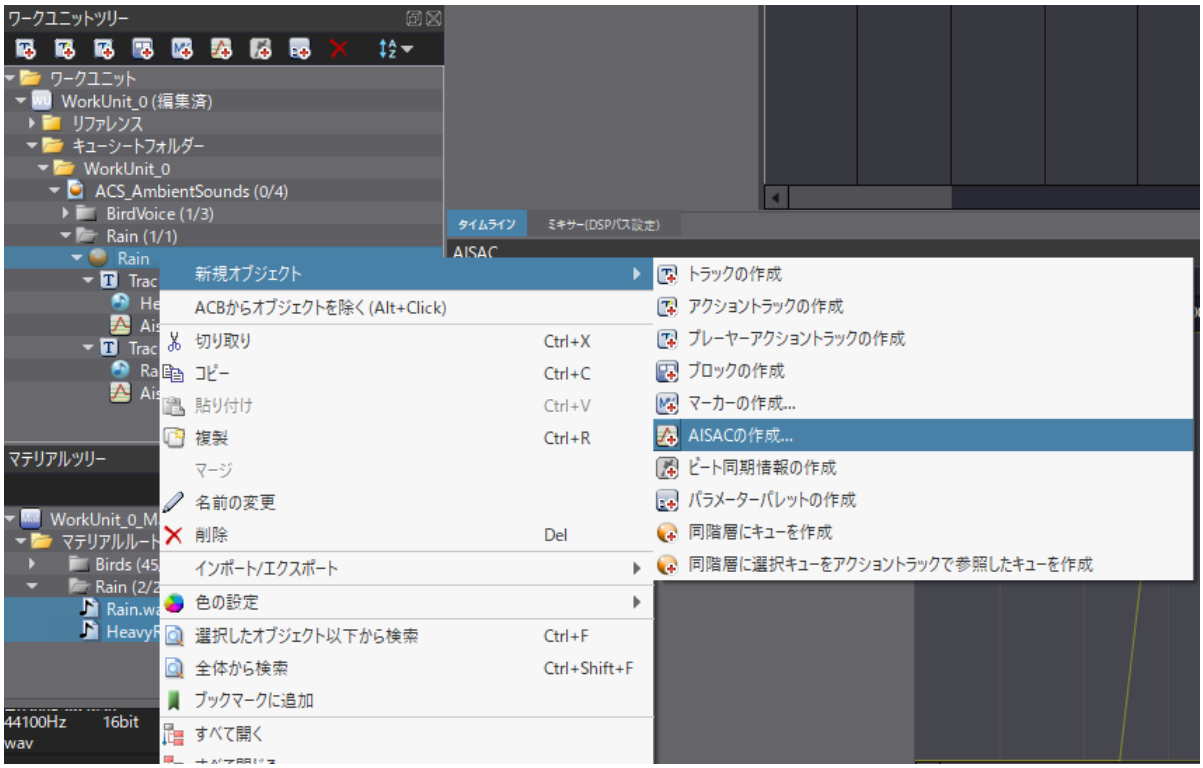
8．Aisac ウィンドウ上で、Track_Rain の Aisac_0 を選択し、グラフを作成していきます。
図のハイライトが入っているようにグラフを作成します。
またデフォルトコントロール値を有効にして 0 で設定します。

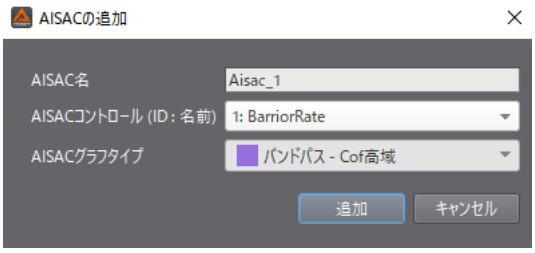


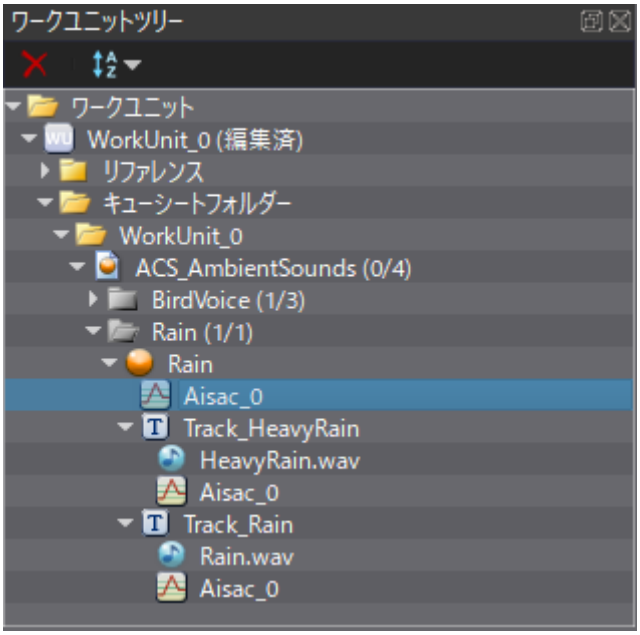
9．Aisac ウィンドウ上で、Track_HeavyRain の Aisac_0 を選択し、グラフを作成していきます。
図のハイライトが入っているようにグラフを作成します。
またデフォルトコントロール値を有効にして 0 で設定します。



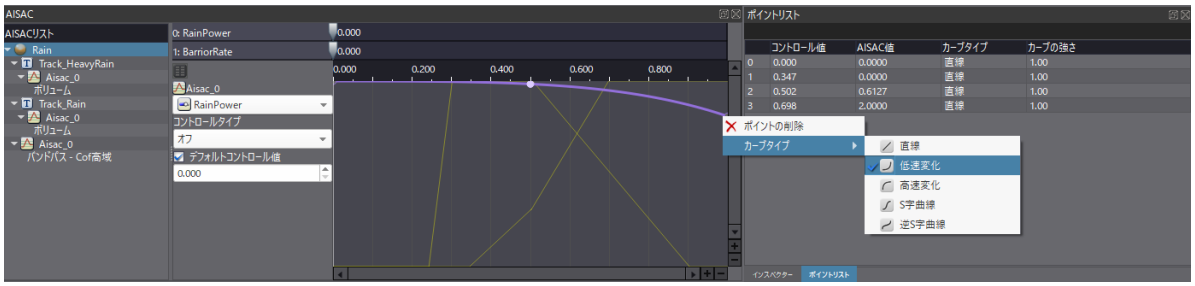
１０．Rain キューを選択して、右クリックから Aisac の作成を行い、
Aisac コントロールに BarriorRate と、Aisac グラフタイプにバンドパス - Cof 高域を指定し
追加ボタンを押します。

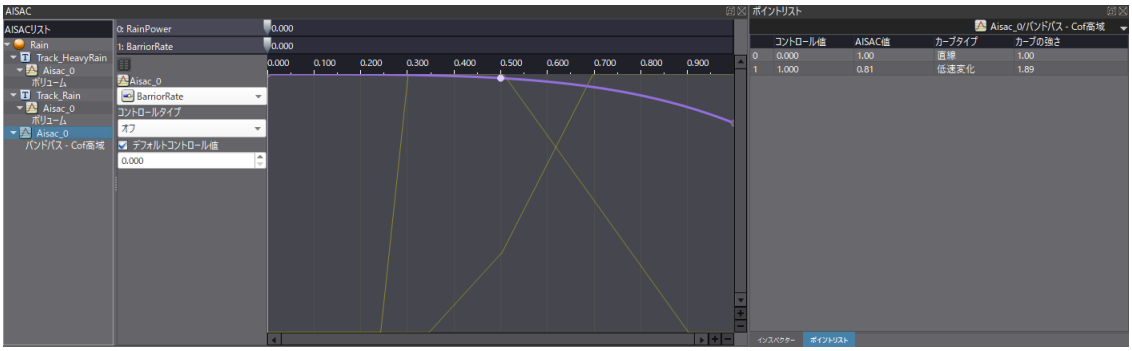




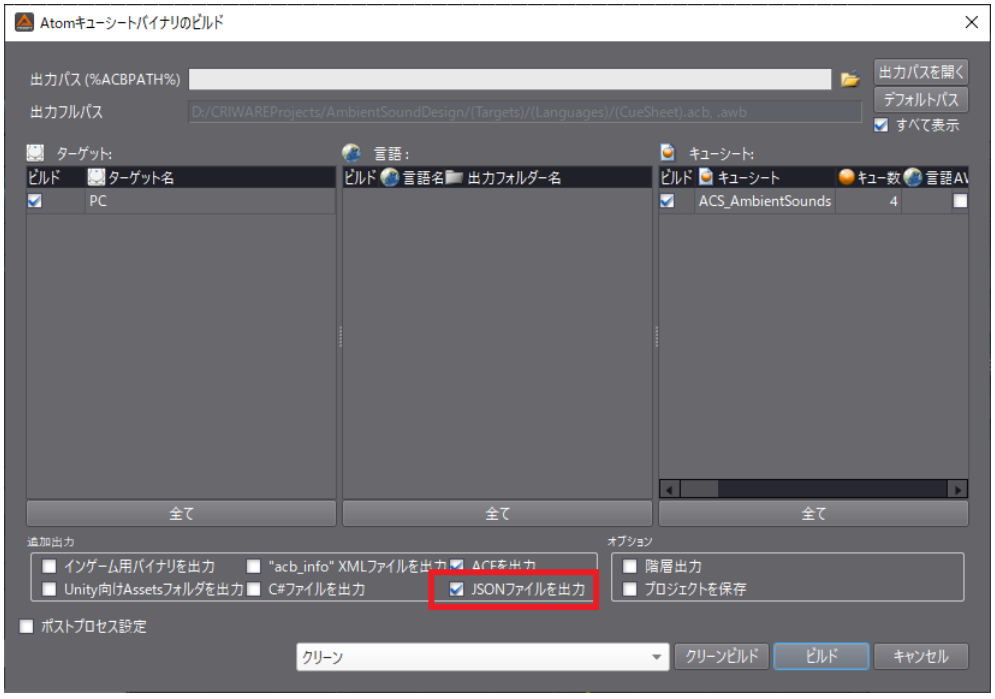


１１．Aisac ウィンドウ上で、ルートの Aisac_0 を選択し、グラフを作成していきます。
図のハイライトが入っているようにグラフを作成します。
またデフォルトコントロール値を有効にして 0 で設定します。
※なお、カーブのキーの上で右クリックすると、カーブタイプを選択できます。図ではカーブタイプを低速変化にしています。





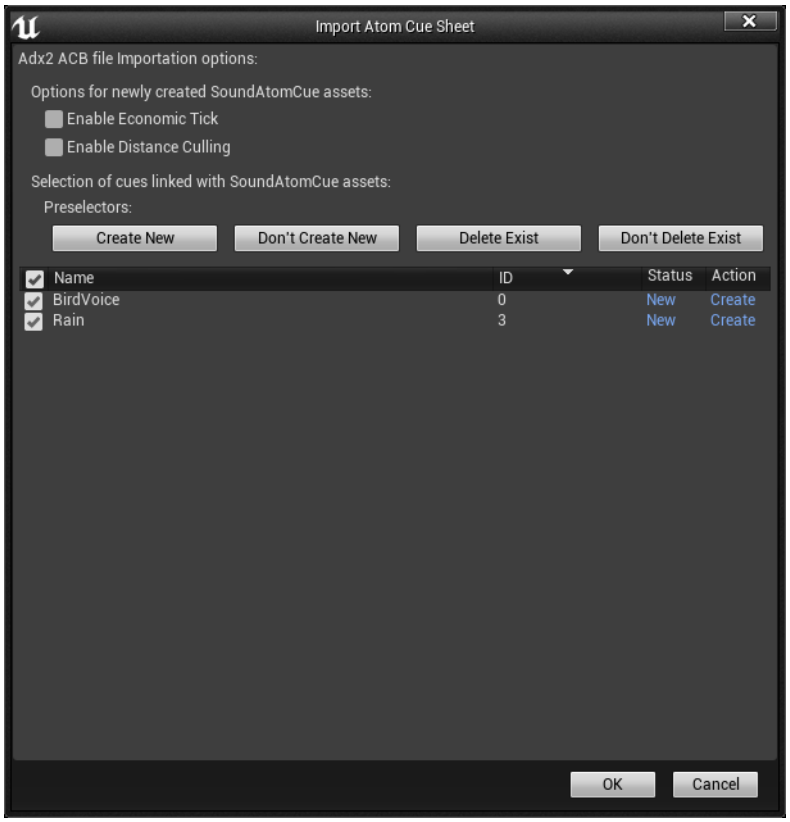
１２．データをリビルドします。
リビルドの際に、Json ファイルを出力にチェックを入れた状態でリビルドします。
すると ACS_AmbientSound.acb と AmbientSound.acf に加えて、
AmbientSound_acf.json という名前のファイルが出来上がります。
この三つのファイルを次項では UE4 にインポートします。

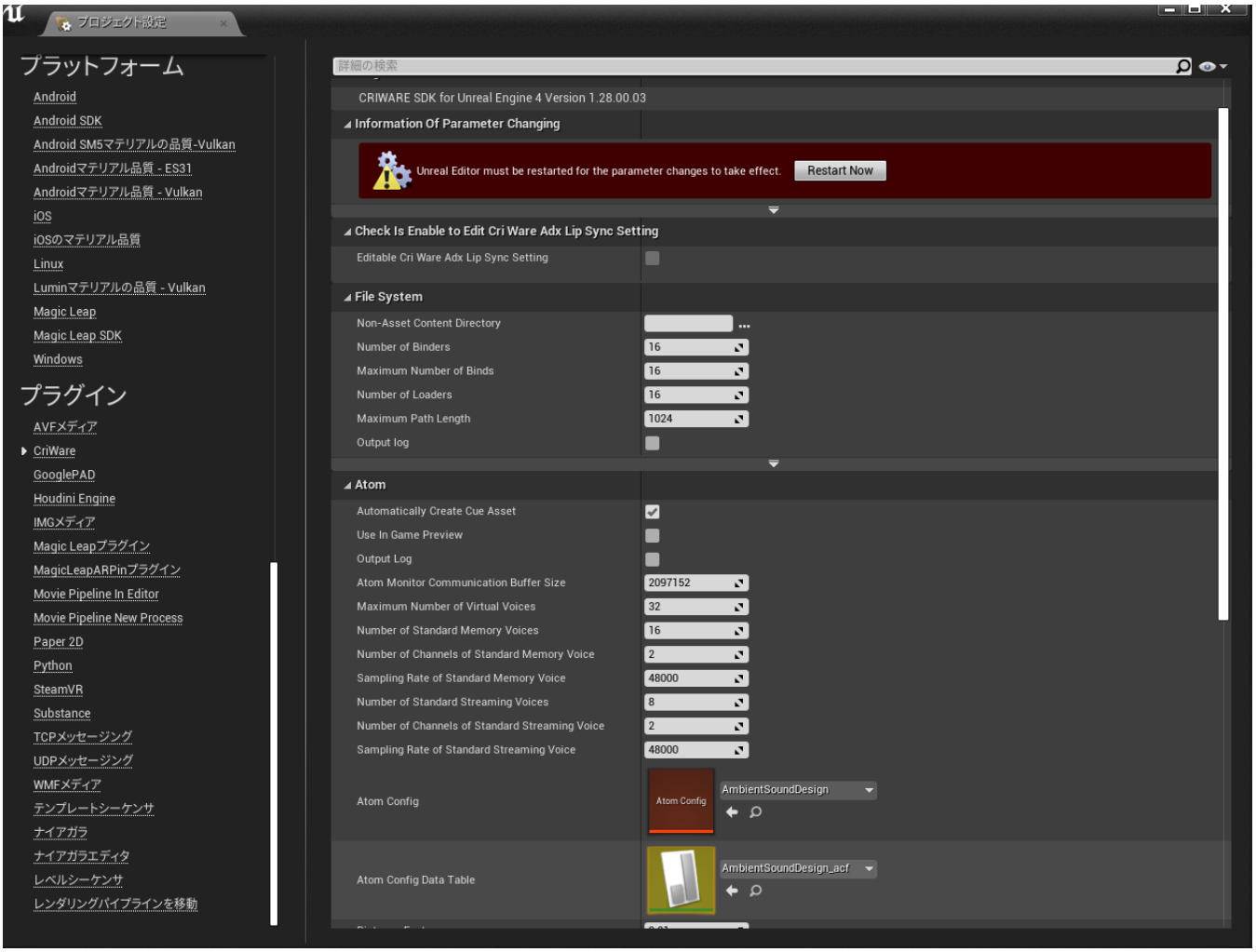


UE4 上での再生

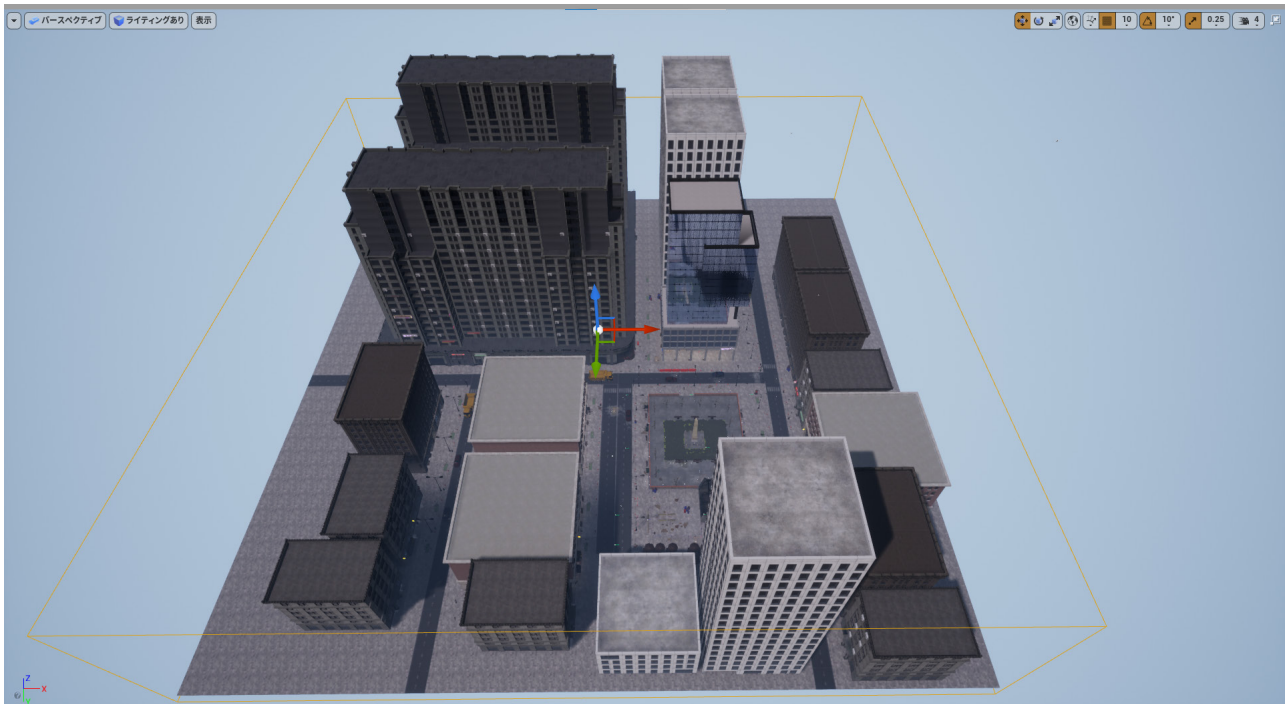
Rain キューをレベル上に配置して、時間経過すると徐々に雨音が強くなっていくよう実装していきます。

1．AmbientSound.acb と AmbientSound.acf と AmbientSound_acf.json をインポート
（既にインポート済みの場合はリインポート）します。
AmbientSound_acf.json 時にはどのデータ型でインポートするかというウィンドウが開くので、
AtomCongigDataTable 型を選んでアセットを作成します。
インポートの後、プロジェクト設定→ CriWare カテゴリ内の AtomConfigDataTable に
AmbientSound_acf アセットをセットし、UE4Editor を再起動します。

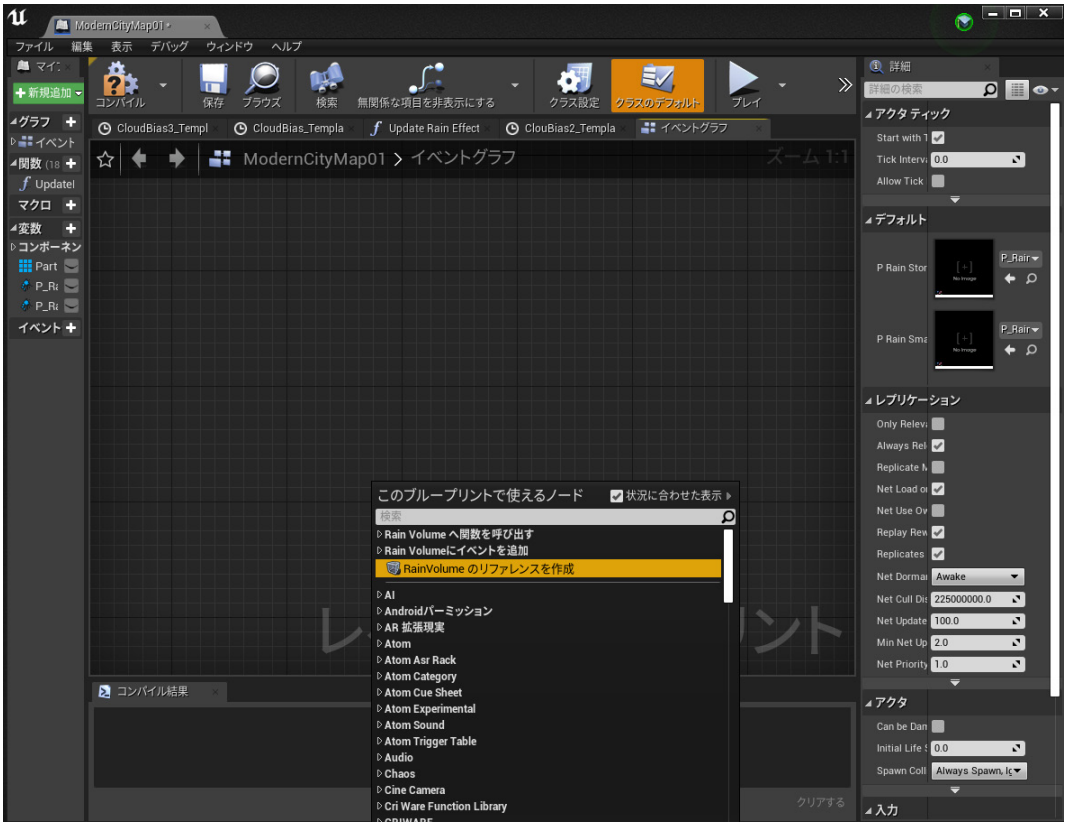


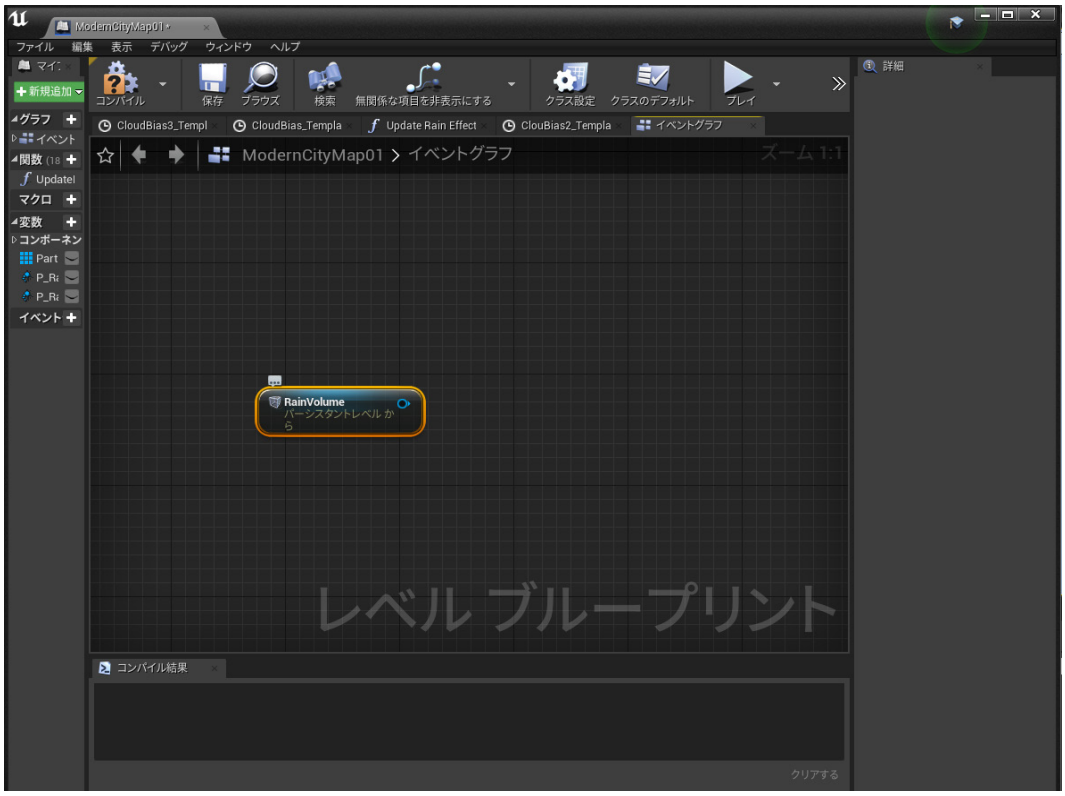

プロジェクト設定
プラットフォーム
Android
Android SDK
Android SM5マテリアルの品質-Vulkan
Androidマテリアル品質 - ES31
Androidマテリアル品質 - Vulkan
iOS
iOSのマテリアル品質
Linux
Luminマテリアルの品質 - Vulkan
Magic Leap
Magic Leap SDK
Windows
プラグイン
AVFメディア
CriWare
GooglePAD
Houdini Engine
IMGメディア
Magic Leapプラグイン
MagicLeapARPinプラグイン
Movie Pipeline In Editor
Movie Pipeline New Process
Paper 2D
Python
SteamVR
Substance
TCPメッセージング
UDPメッセージング
WMFメディア
テンプレートシーケンサ
ナイアガラ
ナイアガラエディタ
レベルシーケンサ
レンダリングパイプラインを移動
詳細の検索
CRIWARE SDK for Unreal Engine 4 Version 1.28.00.03
Information Of Parameter Changing
Unreal Editor must be restarted for the parameter changes to take effect.
Restart Now
Check Is Enable to Edit Cri Ware Adx Lip Sync Setting
Editable Cri Ware Adx Lip Sync Setting
File System
Non-Asset Content Directory
Number of Binders
16
Maximum Number of Binds
16
Number of Loaders
16
Maximum Path Length
1024
Output log
Atom
Automatically Create Cue Asset
Use In Game Preview
Output Log
Atom Monitor Communication Buffer Size
2097152
Maximum Number of Virtual Voices
32
Number of Standard Memory Voices
16
Number of Channels of Standard Memory Voice
2
Sampling Rate of Standard Memory Voice
48000
Number of Standard Streaming Voices
8
Number of Channels of Standard Streaming Voice
2
Sampling Rate of Standard Streaming Voice
48000
Atom Config
Atom Config
AmbientSoundDesign
Atom Config Data Table
AmbientSoundDesign_acf

2．アクターを配置タブから AtomAreaSoundVolume アクターをレベル上に配置し、
RainVolume とリネームします。続いて、Volume のサイズを Player の移動範囲を覆うよう配置します。
サイズの設定はブラシセッティングの X、Y、Z のパラメータを変更することで調整してください。
サイズを指定した後、Sounds のエレメントを一つ追加し、Rain キューアセットをセットします。

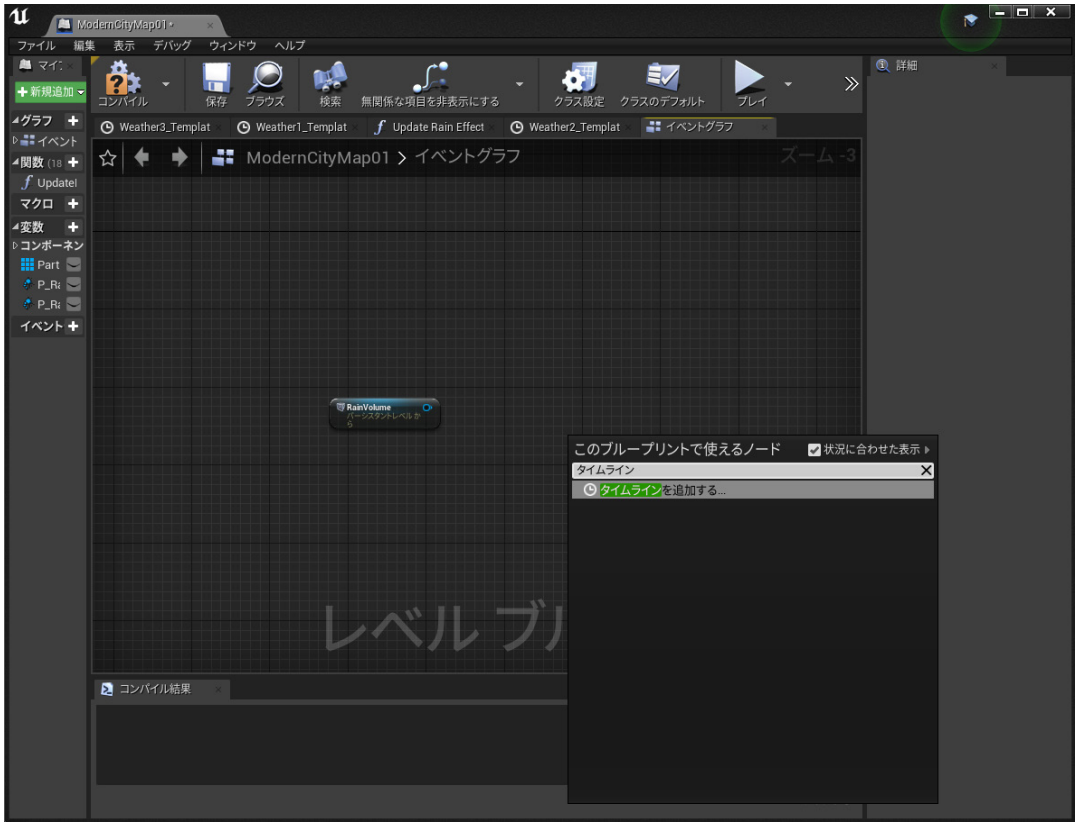


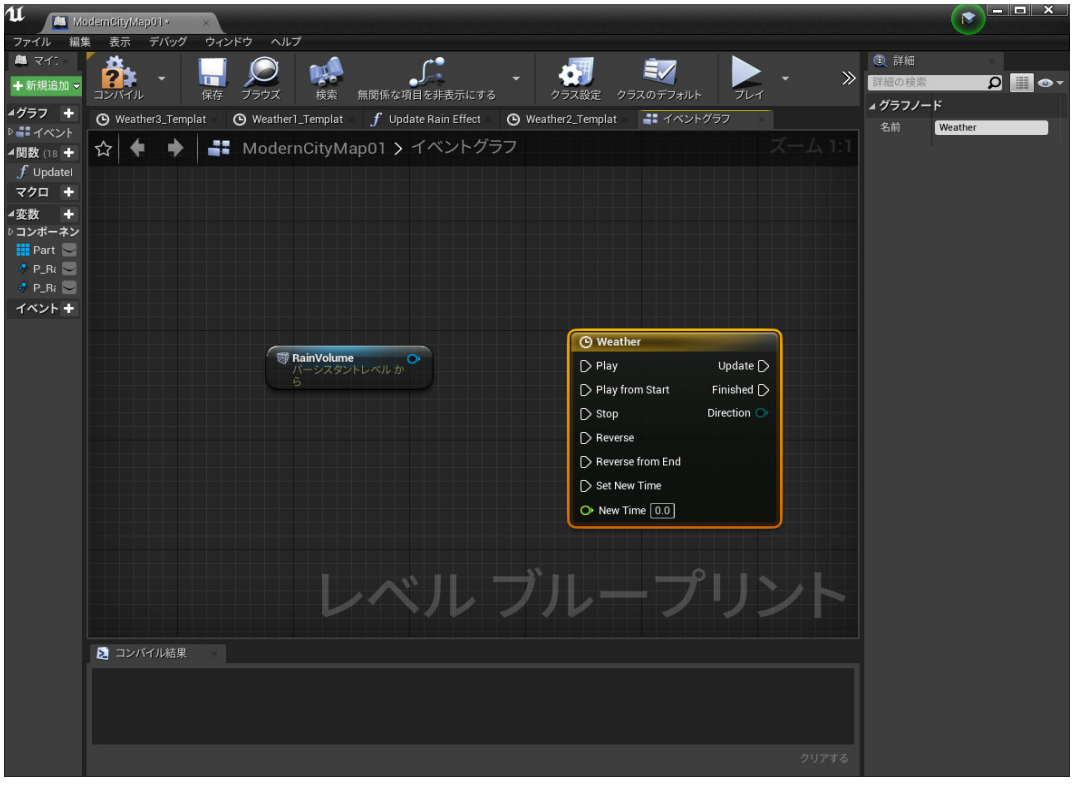
3．レベルブループリントを開き、レベル上に配置した RainVolume アセットを
レベルブループリント上に持っていきます。
RainVolume アクターを選択した状態で、レベルブループリントエディタ上で右クリック
→「RainVolume のリファレンスを作成」を行います。
すると RainVolume アクターの参照ノードが出来上がります。

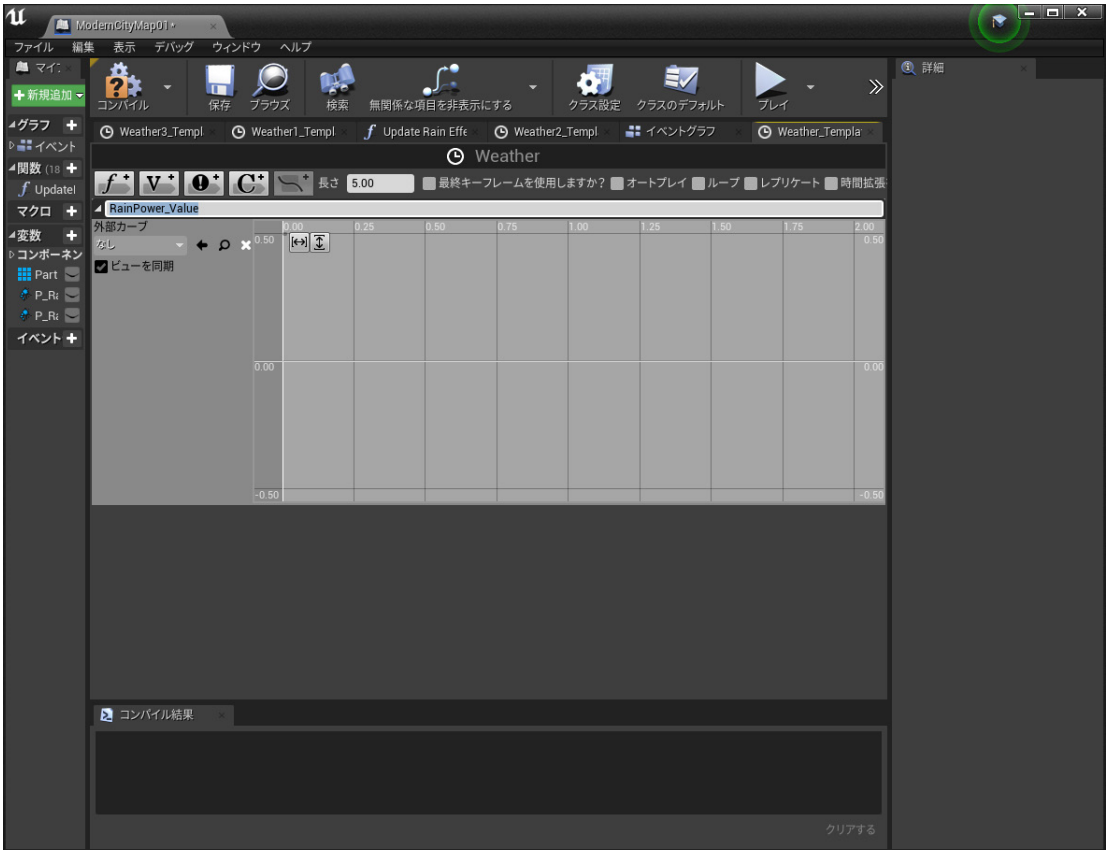


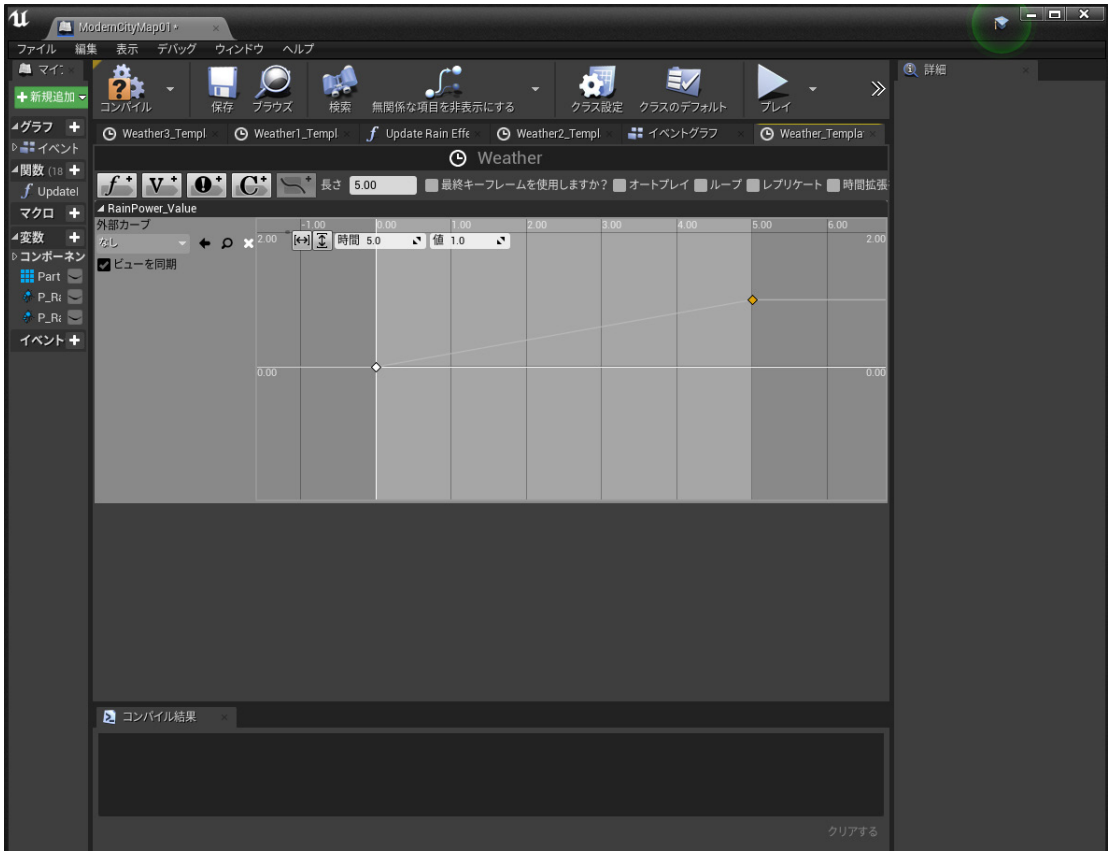


4．「Timeline を追加」でタイムラインノードを作成し、BeginPlay ノードに接続します。
Timeline ノードはダブルクリックすると CurveEdit 画面が開くため、
f+ のボタンを押して float 値カーブグラフを作成します。
この時 Track 名を RainPower_Value とします。続いて、グラフ上で右クリック
→「キーを追加」でカーブを作成していきます。
※事前に天気の移り変わり用のパラメータと、雨の強さ用のパラメータを作成しています。

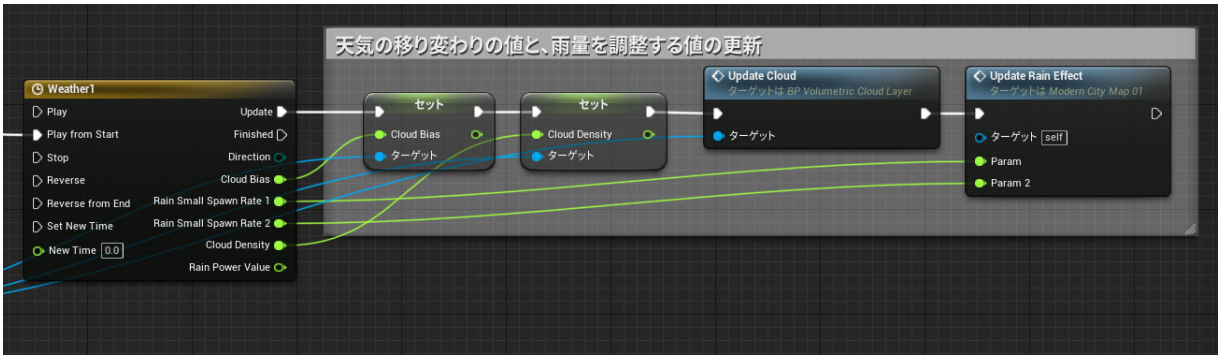




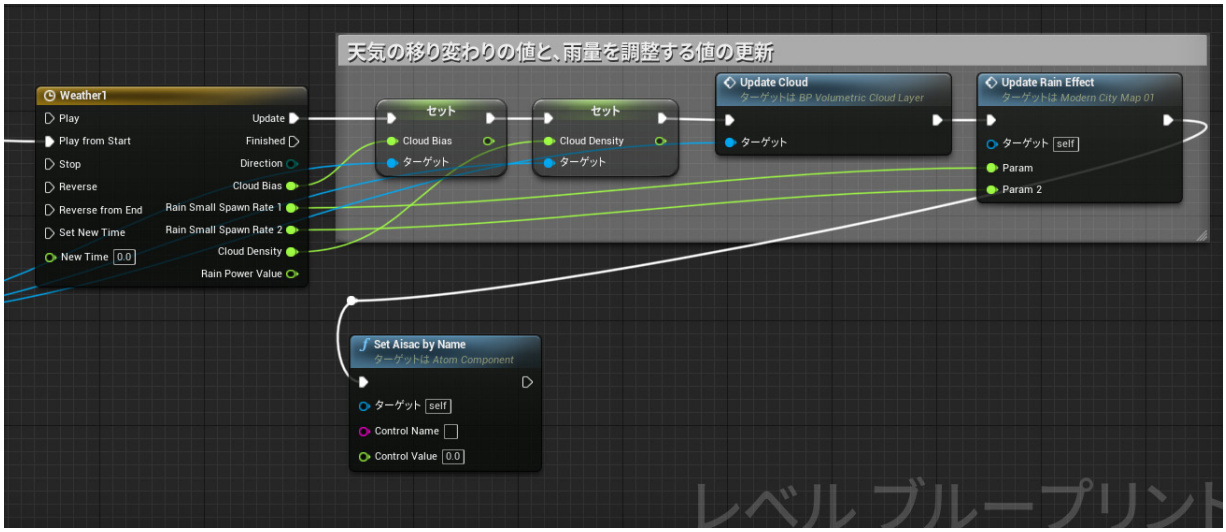

ModernCityMap01
ファイル 編集 表示 デバッグ ウィンドウ ヘルプ
コンパイル 保存 ブラウズ 検索 無関係な項目を非表示にする クラス設定 クラスのデフォルト プレイ
詳細
Weather3_Templ Weather1_Templ Update Rain Effe Weather2_Templ イベントグラフ Weather_Templa
Weather
長さ 5.00 最終キーフレームを使用しますか? オートプレイ ループ レプリケート 時間拡張
RainPower_Value
外部カーブ
なし
ビューを同期
コンパイル結果
クリアする


ModernCityMap01
ファイル 編集 表示 デバッグ ウィンドウ ヘルプ
コンパイル 保存 ブラウズ 検索 無関係な項目を非表示にする クラス設定 クラスのデフォルト プレイ
詳細
Weather3_Templ Weather1_Templ Update Rain Effe Weather2_Templ イベントグラフ Weather_Templa
Weather
長さ 5.00 最終キーフレームを使用しますか? オートプレイ ループ レプリケート 時間拡張
RainPower_Value
外部カーブ
なし
ビューを同期
時間 5.0 値 1.0
コンパイル結果
クリアする

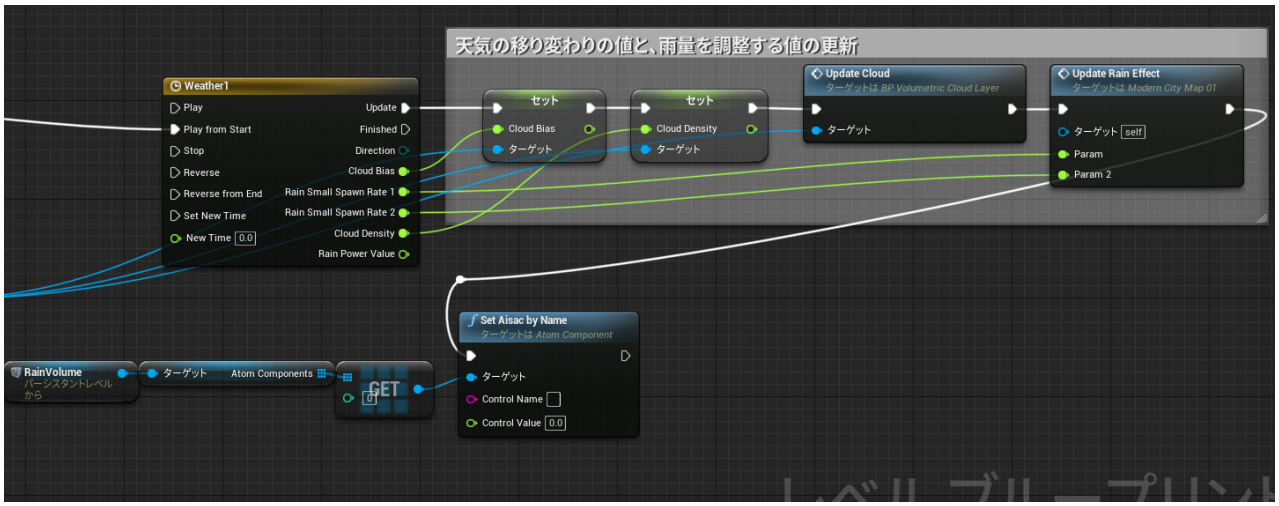
この方法でさらに設定値を増やしていくと、雨のエフェクトと、天気の移り変わり用のパラメータの更新も同時に行う Timeline を作っていくことができます。



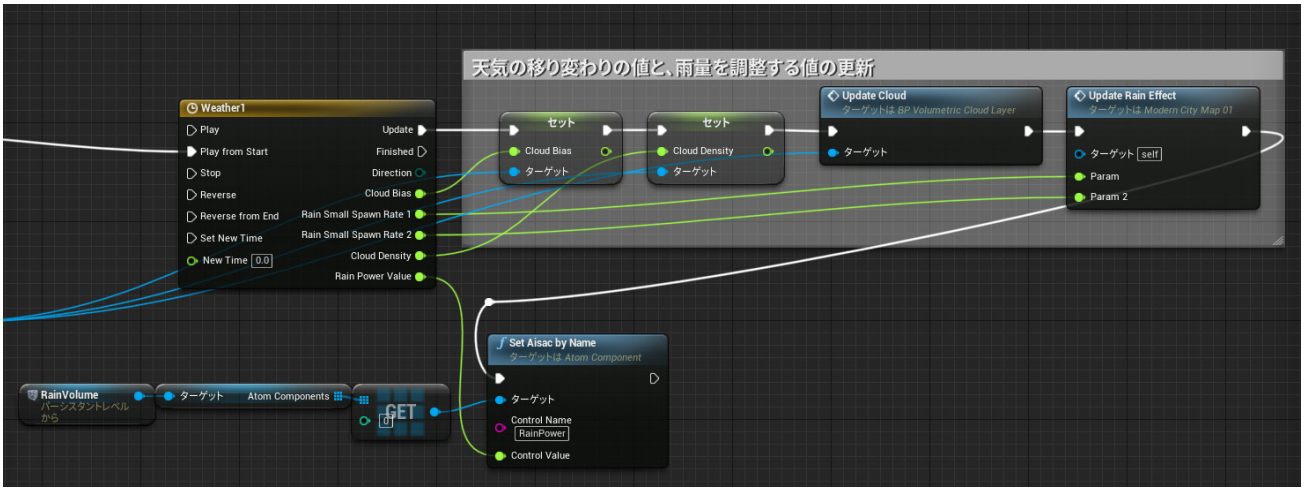
5．レベルブループリントのイベントグラフ画面に戻り、
Timeline ノードの出力ピンの Update に SetAisacControlByName ノードを接続します。



6．RainVolume アクターの出力ピンをドラッグし、
GetAtomComponents ノードをノード検索からレベルブループリント上へ配置します。
AtomComponents ノードの出力ピンであるアレイピンををドラッグして、
「コピーを取得」で Get ノードを作成します。



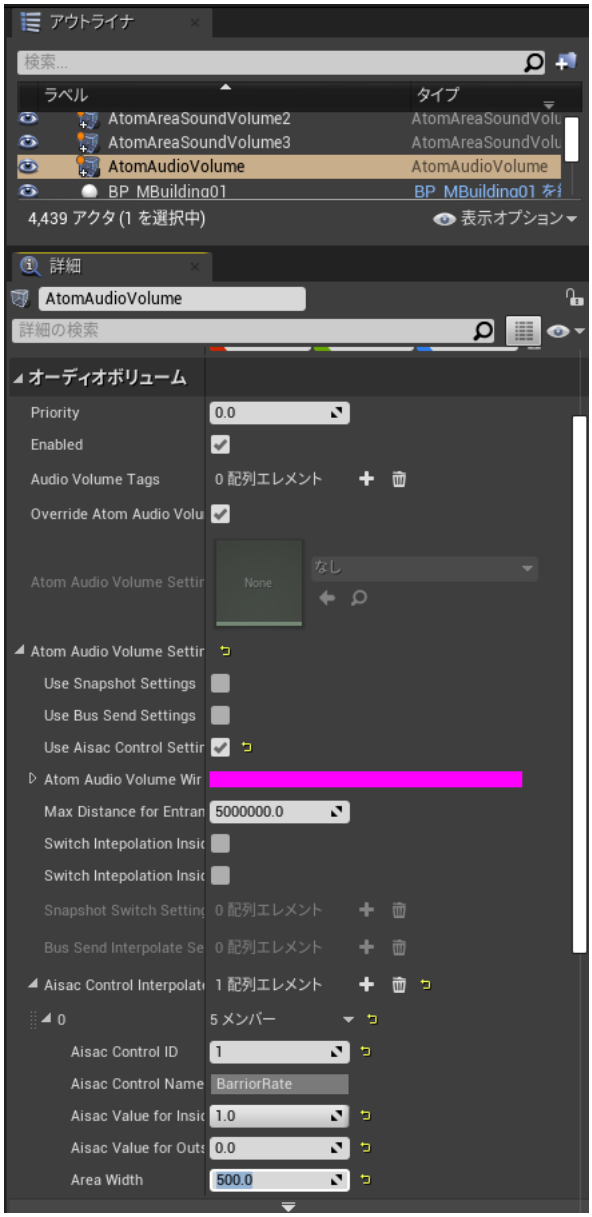
7．Get ノードの出力ピンを SetAisacControlByName ノードのターゲットに接続し、
ControlName に RainPower を指定し、ControlValue に Timeline の RainPower_Value に接続します。



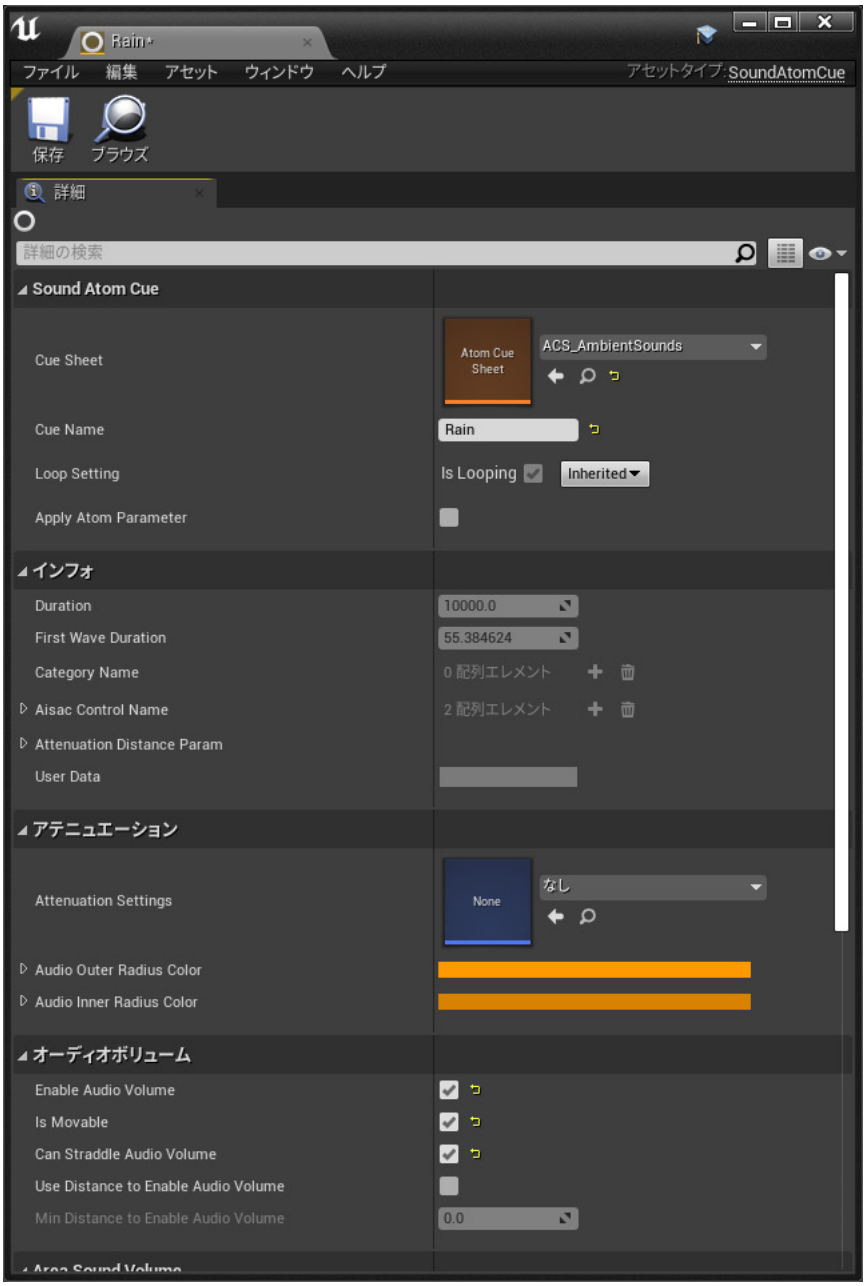
8．次に屋根のある建物に入ったら、音がこもるような音に変化するようにしていきます。
今回はレベル上の公園をまっすぐ歩いた先にあるハンバーガ屋の建物に入ったら、
音がこもるような音に代わるようしていきます。
そのためにはまず、レベルブループリント上に AtomAudioVolume アセットを配置します。
建物で Player が入った際に屋根がある部分全部を一つの AtomAudioVolume で覆います。
図では柱のあるエリアを囲うように Volume を配置しています。
変更の際にはブラシセッティングの X、Y、Z で変更します。

※ AtomAudioVolume とは？
Aisac による変化や、エフェクトの変化をリスナーやサウンドの Volume 内外への移動で
動的に適用するための機能です。
本機能は開けた空間から洞窟などの空間的な状況が変わるような場合に、
AtomAudioVolume を洞窟に配置することでサウンドが Volume 内で再生している時にリバーブをかけたりして、
洞窟の音響空間をプログラムレスで表現することができます。
実用例：
・建物内外への移動

9．AtomAudioVolume アセットの詳細パネル中のオーディオボリュームカテゴリの内部を編集していきます。
まず、UseAisacControlSettings を有効にします。
次に、AisacControlInterpolateSettings のエレメントを一つ追加します。
作成されたエレメント内の AisacControlID を 1 に変更します。
これにより AisacControlName が BarriorRate となります。
AisacValueForInside の値を 1.0 に、AisacControlValueForOutSide の値を 0.0 に、
AreaWidth を 500 に設定します。



１０．コンテンツブラウザ上の Rain キューアセットをダブルクリックして、アセットエディタで開きます。
「EnableAudioVolume」、「IsMovable」、「CanStraddleAudioVolume」、「UseAreaSoundVolume」の
フラグを有効にします。



１１．PlayInEditor で実行します。これにより時間の経過で雨が強くなり、
AtomAudioVolume を配置した領域にプレイヤーが侵入すると、
音がこもったように聞こえるようになります。

chapter 6 雨音と鳥の鳴き声の連動

今までの章で説明したことを踏まえ、サウンドをさらに作りこんでみましょう。

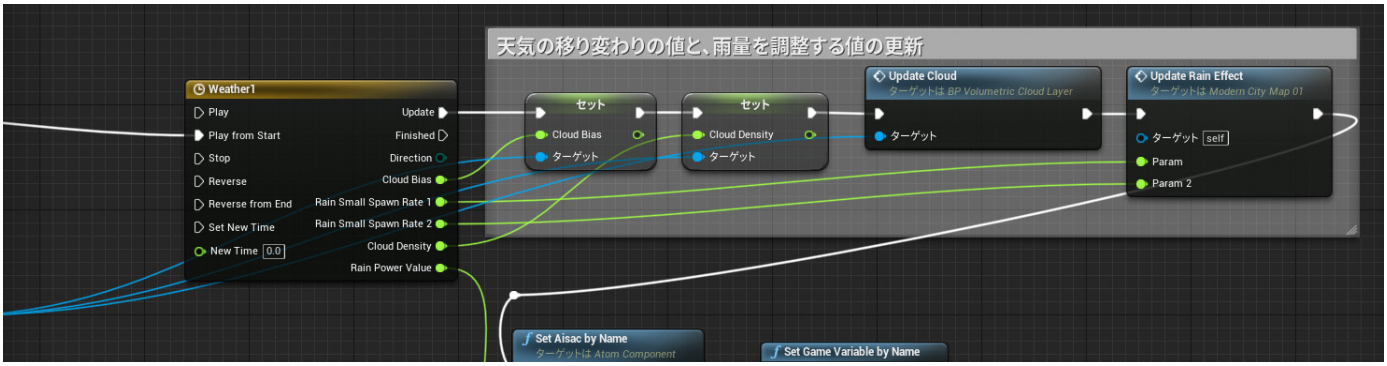
雨音と鳥の鳴き声の連動

現時点で Rain キューと BirdVoice キューが UE4 上で、
それぞれ独立して音の変化を適用させられるようになりました。

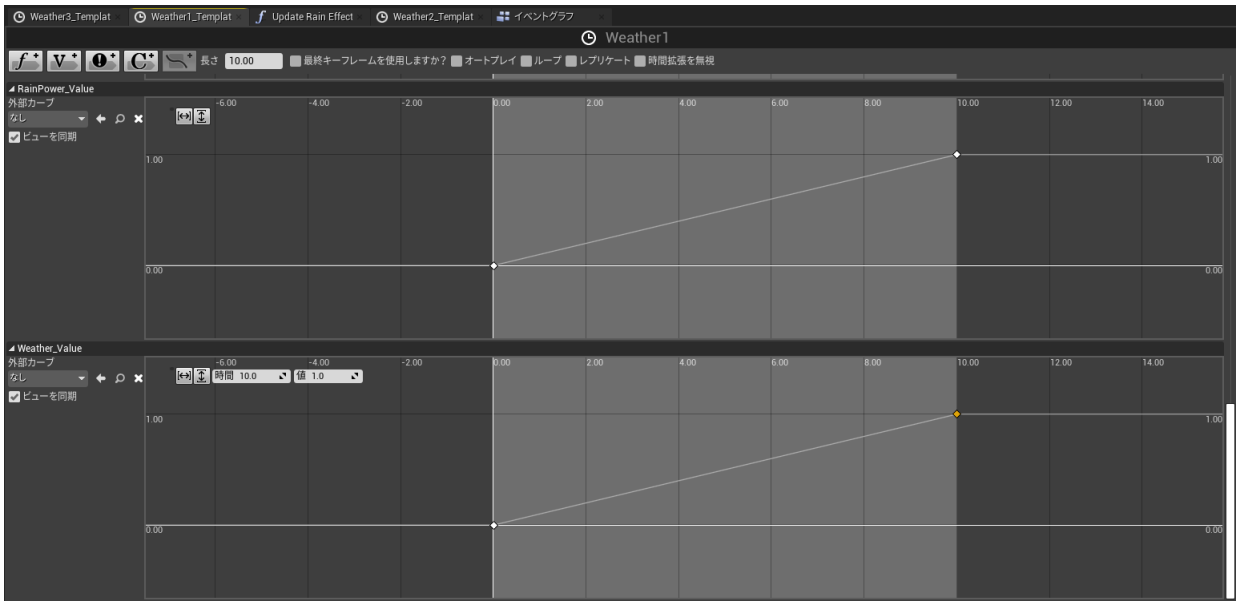
しかし、木々から漏れる鳥の声の仕様にある
「・開始時刻と再生タイミングランダムを別に設定したトラックを複数用意し、
ゲーム変数を使って雨音の強さ（AisacControl 値、RainPower）に応じた段階的な再生頻度変化を与える」が
未だ達成されていません。
次は Rain の AisacControl 値（RainPower）の変化に合わせて、
BirdVoice の再生頻度が変わっていくように Blueprint の内容を書き換えていきます。

1．レベルブループリント上のキー入力によるゲーム変数の変更処理をすべて削除します。

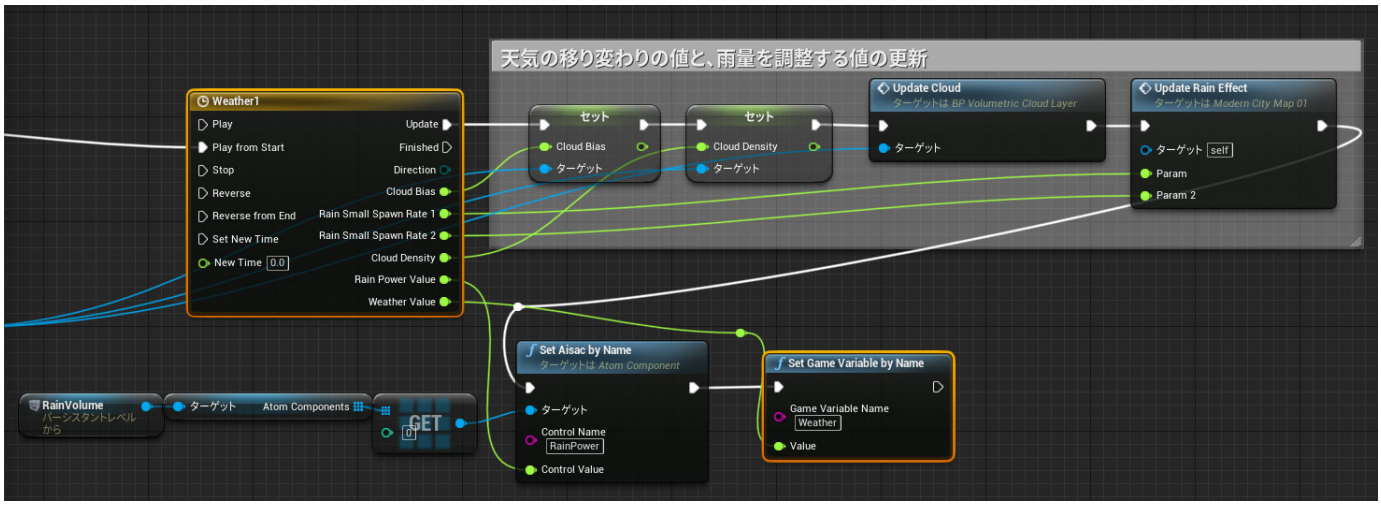
2．レベルブループリント上に設定した SetAisacControlByName の出力ピンに
SetAtomGameVariable のノードを接続します。
GameVariableName ピンに「Weather」を指定します。



3．Timeline ノードに新しく Weather_Valuer 変数を追加します。グラフのカーブは、
今回は直線で 0 ～ 1 に線形に変化していくものにします。



4．カーブの設定が終わったら、イベントグラフウィンドウに戻り、
Timeline ノードの Weather_Value ピンと SetGameVariableByName ノードの Value ピンを接続します。



5．PlayInEditor でアプリケーションを実行します。ここまでの処理により、時間経過に応じて雨音が強くなり、
それに応じて鳥の鳴き声の再生頻度が段階的に減っていくということが実現できます。

chapter 7 まとめ

今までの章で説明したことを踏まえ、サウンドをさらに作りこんでみましょう。

本書では「プロシージャルデザイン」「パラメータコントロールデザイン」の 2 種類のサウンドデザインについて触れつつ、実際のアンビエントサウンドのデザインフローをゼロから解説してきました。

他のどのようなデザインのサウンド演出を実装するにしても、
多くの場合は以下のステップを踏む必要があります。

1. サウンド演出の要件を定める (ex. ゲーム内時刻に応じて雨音が強まる)
2. 要件を実現するにはどのような仕様が必要か検討する
   (ex. ゲーム内時刻のような数値情報に連動して、音のボリュームを変化させる)
3. 2 で考えた仕様を実装するための ADX2 の機能を選択する (ex. AISAC)

今回は説明を簡単にするために利用する ADX2 の機能を絞って説明してきましたが、
BusEffect などまだまだ使ってない機能はたくさんあり、
これらの機能を利用することでさらに良いサウンド演出が実現できるようになってきます。

そのためにも ADX2 の機能をたくさん触り、どのような機能があるのかをまず知り、
どの機能とどの機能を組み合わせるとどんな演出ができるのか、などぜひ試行錯誤していただければと思います。

参考資料

・「アンビエントおよびプロシージャル サウンド デザイン」
https://www.unrealengine.com/ja/onlinelearning-courses/ambient-and-procedural-sound-design

・「ADX2 ポータル UE4 tutorial」
https://game.criware.jp/learn/tutorial/ue4/

・「ウェビナー」
https://criware.info/webiner2/

書籍内での使用アセット

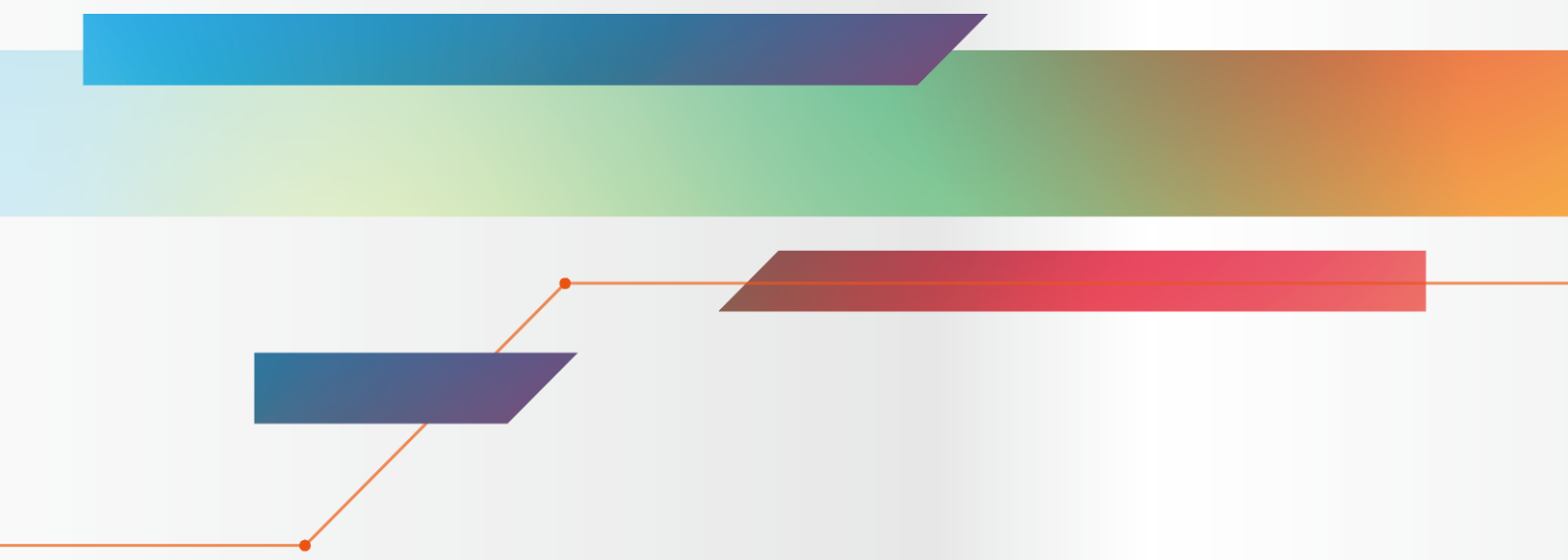
■森の画像
Meadow - Environment Set
https://www.unrealengine.com/marketplace/en-US/product/meadow-environment-set

■町の画像
Modern City Downtown with Interiors Megapack (ModularUrban Buildings)
https://www.unrealengine.com/marketplace/en-US/product/modern-city-downtown-with-interiors-megapack

発行日　2021 年 2 月 1 0 日
著者　上田 雄太（うえだ ゆうた）
発行人　押見 正雄
発行所　株式会社 CRI・ミドルウェア
１５０－０００２　東京都渋谷区渋谷　１－７－７　住友不動産青山通ビル９階
https://game.criware.jp/

UE4×ADX2
Beginners Book

UE4×ADX2 導入マニュアル



CRIWARE®