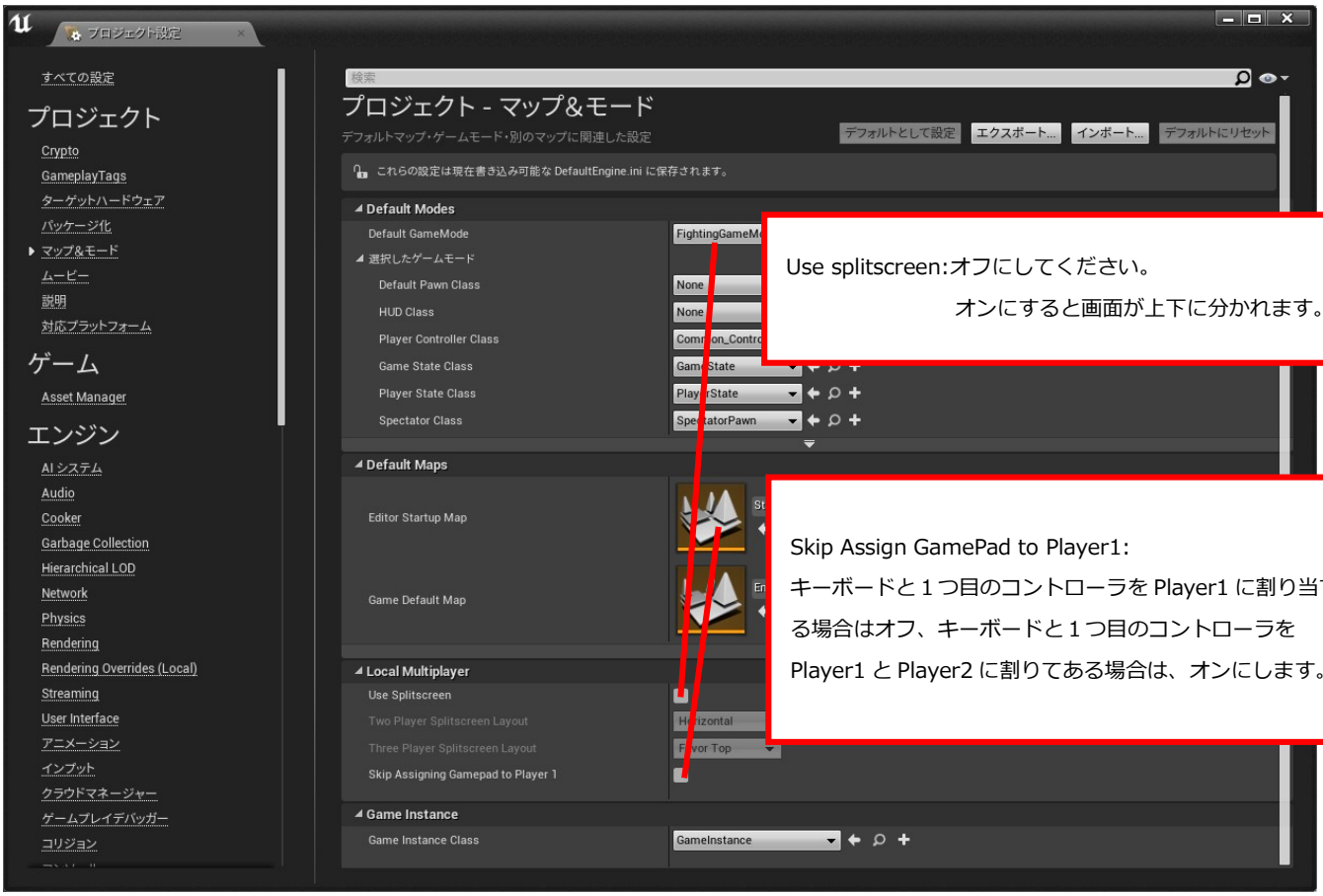
Unreal Engine 格闘ゲーム
テンプレートプロジェクト マニュアル Ver.3.0

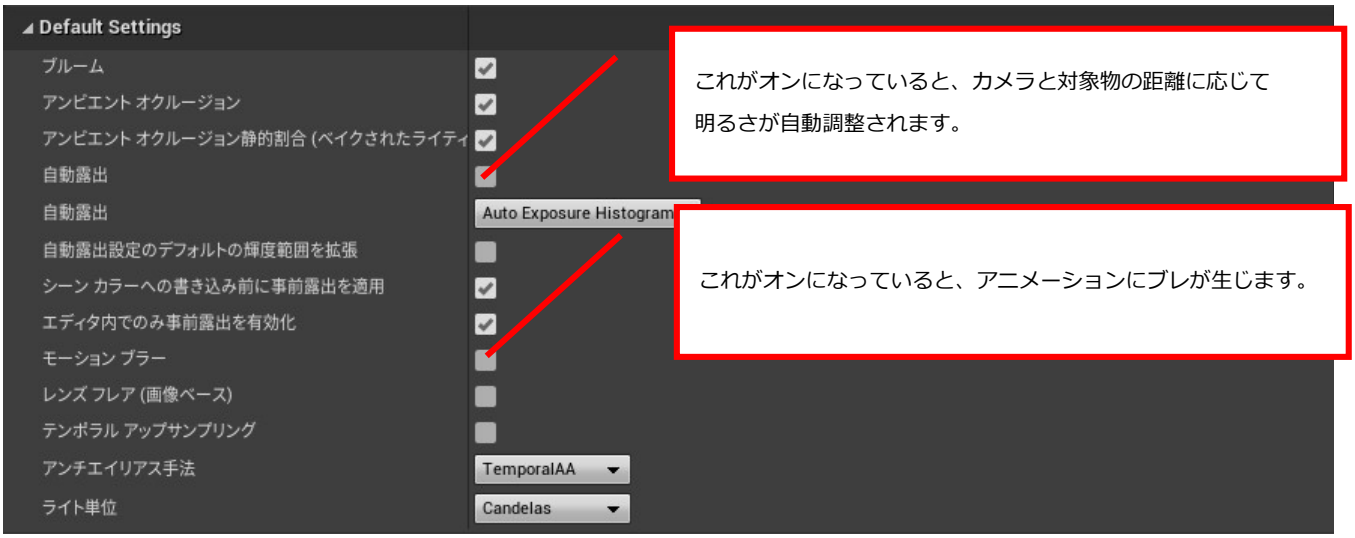
1．プロジェクト設定について（Ver.3.0 の変更点は青字で記載）

プロジェクト設定にあるマップ＆モード内のローカルマルチプレイヤーの設定を下の画像を見て確認してください。



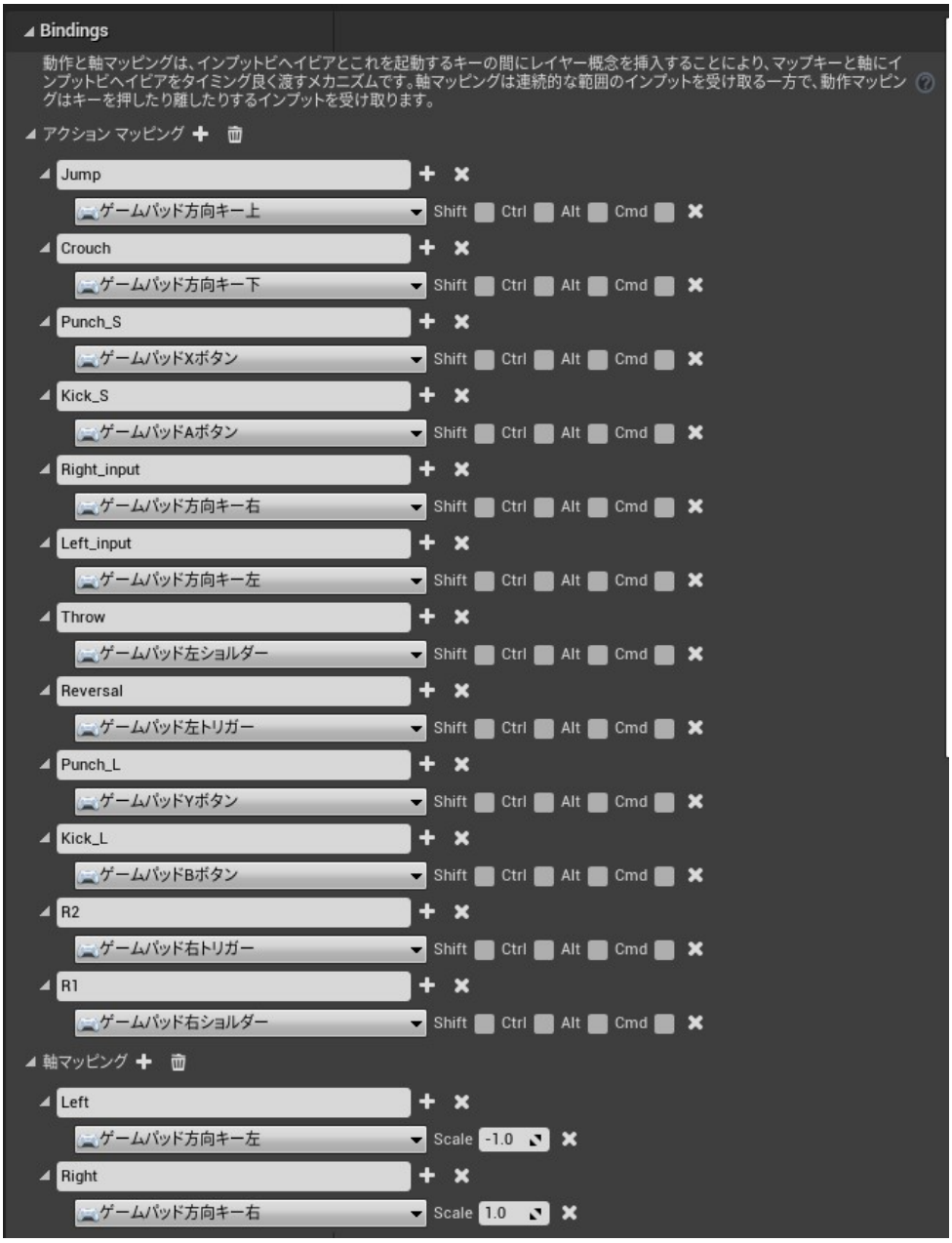
次にエンジン→レンダリングの中にある自動露光をオフにします。

更にモーションブラーをオフにします。



次に、エンジン→インプットの中にあるコントローラの入力設定を以下の画面通りに設定します。

マッピングの名前を間違えると正しく動作しませんので注意願います。



最後にレベルエディタ→プレイにて実行時のウインドウサイズを指定します。

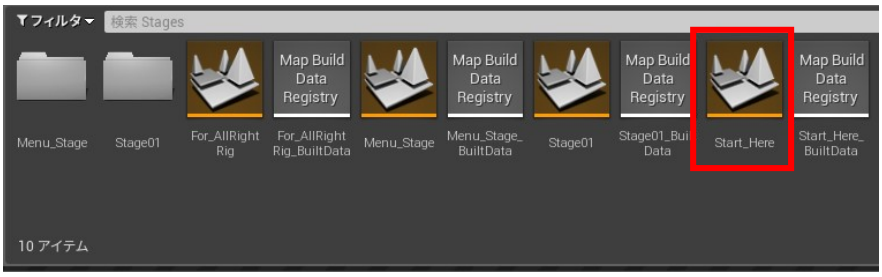
縦横の比が 16:9 になるサイズを入力します。

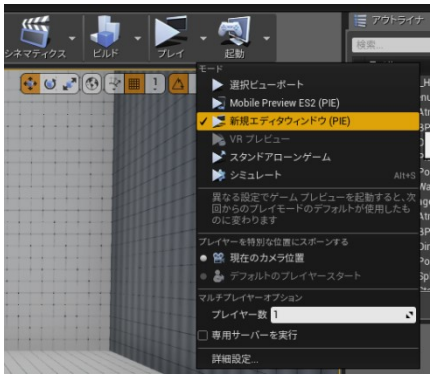
（例:1280*720 / 1600*900 / 1920*1080）

これでプロジェクト設定は完了です。

プロジェクト内のコンテンツにある Stage フォルダの Start_Here ファイルを開いていることを確認したら、実行を試してください。

（新規エディタウインドウでサイズが反映されたか確認してください。）





2．ゲーム中の操作方法について

操作方法の説明はプレイステーションのコントローラのボタンで説明します。

■タイトル画面

○ボタンで次へ進みます。

■ファイティングモードセレクト

1on1 か 3on3 を方向キーで選択し、○ボタンで決定です。×ボタンでタイトル画面へ戻ります。

■モードセレクト

VS CPU（1on1 のみ）か VS Player を選択します。×ボタンでファイティングモードへ戻ります。

■ステージセレクト

方向キーで選択し、○ボタンで決定です。×ボタンでモードセレクトへ戻ります。

■キャラクターセレクト

方向キーで選択し、○ボタンで決定です。

3on3 の場合は、3 人選択しますが、同じキャラクターを複数回選択することはできません。

×ボタンで選択をキャンセルできます。

■バトル中の操作方法

方向キー：左右で前後移動、上はジャンプ、下はしゃがみ

□ボタン：弱パンチ

△ボタン：強パンチ

×ボタン：弱キック

○ボタン：強キック

L1 ボタン：投げと投げ抜け

L2 ボタン：ガード中の反撃（SP バー 1 本以上必要）

R1 ボタン： 2 番目のサブキャラクターによるアシスト攻撃（3on3 のみ）

R1 ボタン＋方向キー前： 2 番目のキャラクターとチェンジ（3on3 のみ）

R2 ボタン：スウェー（1on1 のみ）、 3 番目のサブキャラクターによるアシスト攻撃（3on3 のみ）

R2 ボタン＋方向キー前： 3 番目のキャラクターとチェンジ（3on3 のみ）

□必殺技（キャラクター右向き時）

236+パンチボタン：Fireball（飛び道具）

214+パンチバタン：Flash Straight（空中可）

623+パンチボタン：Flash Straight（対空）

236236+パンチボタン：Iceball（SP バー満タン時）

□チェーンコンボ（サンプル）

立ち状態で×、□、△ボタンを一定のタイミングで順番に押す。

□エアリアルコンボ（サンプル）

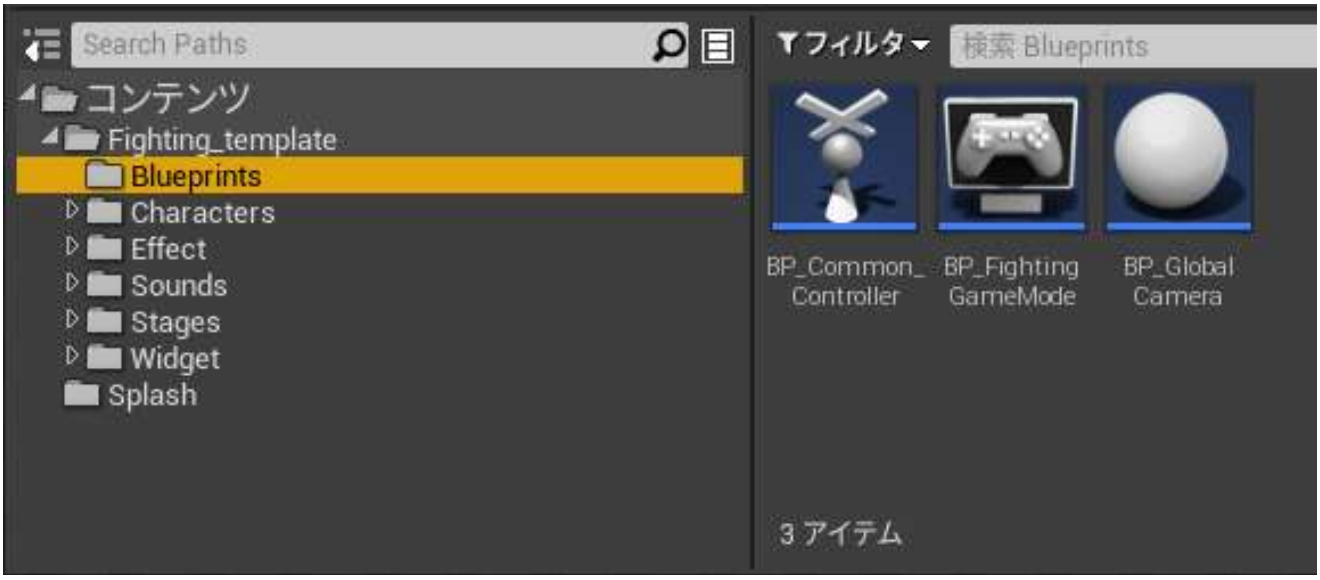
チェーンコンボの△ボタンの攻撃ヒット時に方向キーの上を入力すると、相手を上に打ち上げた後、追尾するので、×、□、△ボタンを一定のタイミングで順番に押す。

△ボタンの攻撃ヒット時に、キャンセルで Flash Straight の空中版を使用することが可能。

3．フォルダ構成とファイルについて

以下フォルダとファイルの概要を説明します。

■Blueprints フォルダ



BP_CommonController：このプロジェクトのメインプログラムファイルです。
プレイヤーの入力を受け取り、キャラクタや各種ゲージの制御を行います。
Player2 の制御は CommonController-1 という形でファイルが生成され、
Controller_ID という変数の値により、Player1 と 2 の識別が行われます。

BP_FightingGameMode：このプロジェクトの初期設定を行うファイルです。
基本的に設定を変更する必要はありません。

BP_GlobalCamera ：カメラをコントロールしているプログラムファイルです。

■Character フォルダ

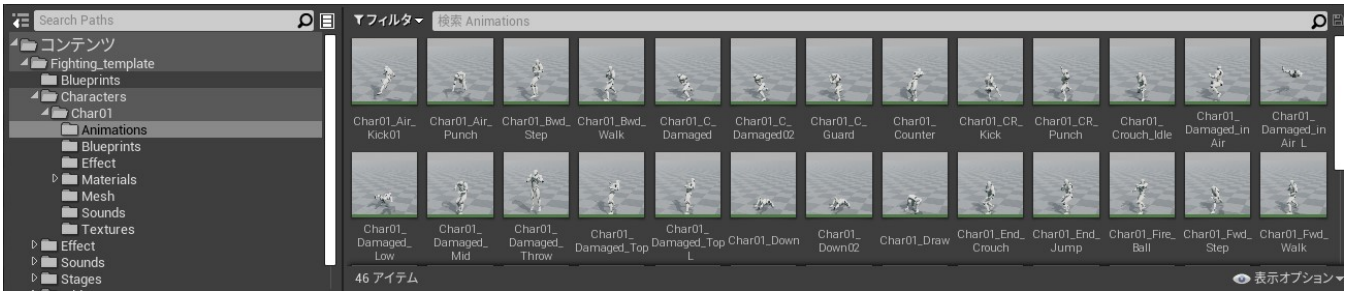
□Char01 フォルダ

キャラクタ毎の専用ファイルが保存されています。

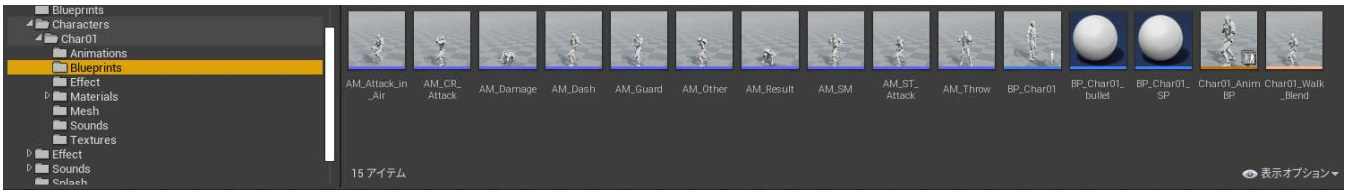
キャラクターを追加する際は Char02、Char03 とフォルダを増やしてください。

□Animations

キャラクタのアニメーションが保存されています。



□Blueprints



・ BP_Char01

キャラクタの攻撃・ダメージの判定 BOX の設定や相手の Capsule component とオーバラップした場合の処理を記載しています。

・ Char01_AnimBP

アニメーションの再生を制御しています。

・ BP_Char01_Bullet

必殺技使用時に発生するアクター（FireBall）を制御しています。

・ BP_Char01_SP

SP ゲージがフルの時に使用できる必殺技使用時に発生するアクター（IceBall）を制御しています。

・その他アニメーションモンタージュ

立ちやしゃがみなどキャラクタの状態毎にアニメーションモンタージュを設定しています。

このファイル内で、通知を利用して攻撃判定の発生・消失のタイミングを指定しています。

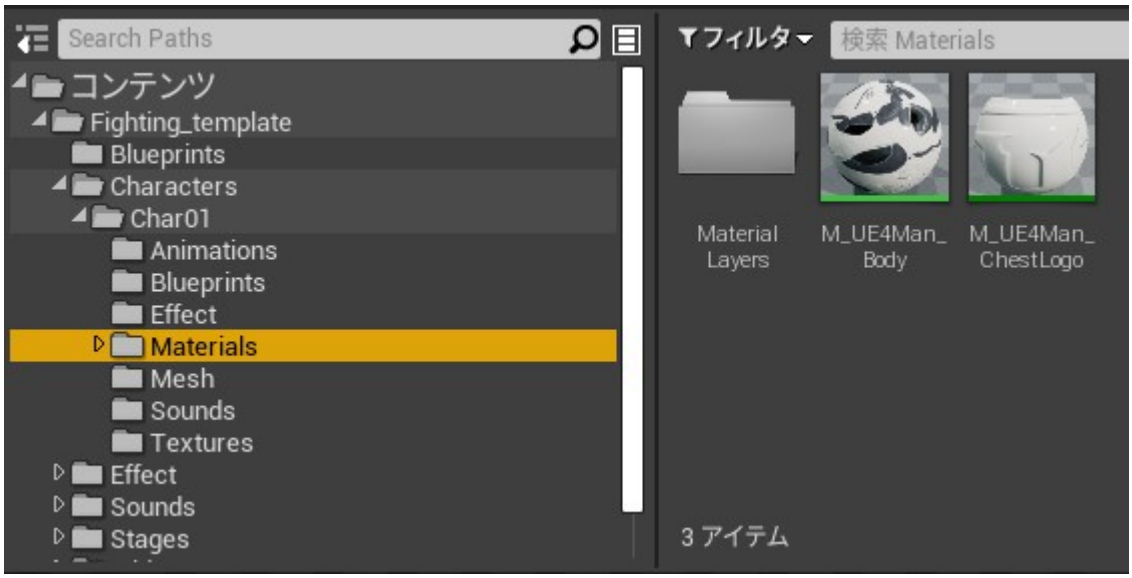
・ Char01_Walk_Blend（ブレンドスペース 1D）

待機状態と前進、後退のアニメーションを制御しています。

・ Effect/Sounds フォルダ

キャラクタ専用必殺技用のパーティクルやサウンドが保存されています。

・ Materials/Mesh/Textures フォルダ



UE4 標準キャラクターであるグレーマンの素材が保存されています。

■Effect フォルダ

ヒットやガードしたときのパーティクルとその素材が保存されています。

■Sounds フォルダ

ヒット時やウィジェットで使用するサウンドが保存されています。

■Stages フォルダ

ステージの素材やレベルのファイルが保存されています。

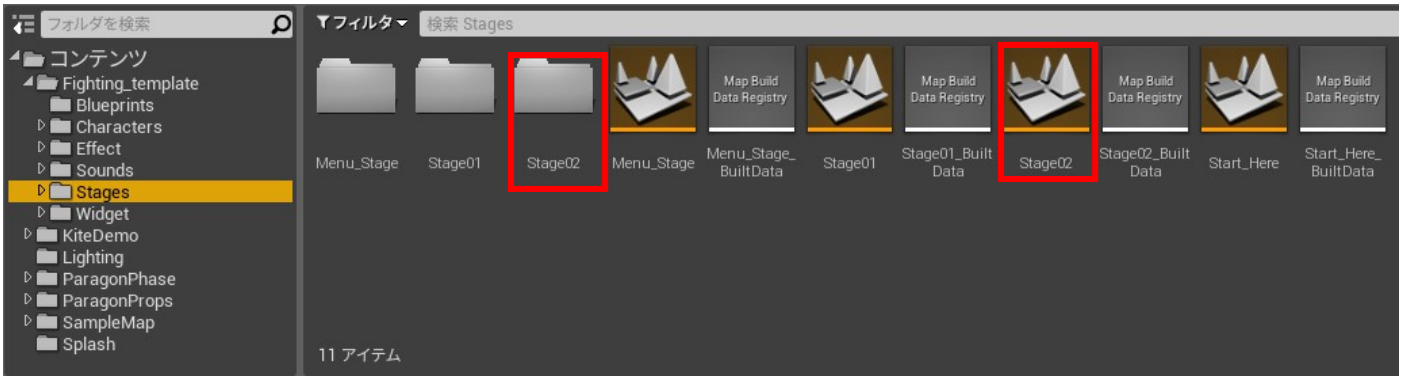
■Widget

メニュー画面等で必要なテクスチャを制御する Blueprint が保存されています。

4．ステージの追加方法

Stages フォルダに素材を格納するフォルダとレベルファイルを追加します。

ファイルとフォルダの名称は、Stage02、Stage03、Stage04 ・・・と追加してください。



Stage02 のレベルファイルを開き、レベルを作成します。

ここでは、事例として、BP_Sky_Sphere と DirectionalLight を追加しておきます。

次に地面となる Plane を追加しますが、壁があるレベルにする場合は、Plane の面積を 20m*20m に、

壁がないレベルにする場合は、面積を 60m*60m にすることをお勧めします。

（ここでは 60m*60m の Plane を追加します。）

次に、座標軸の確認をします。

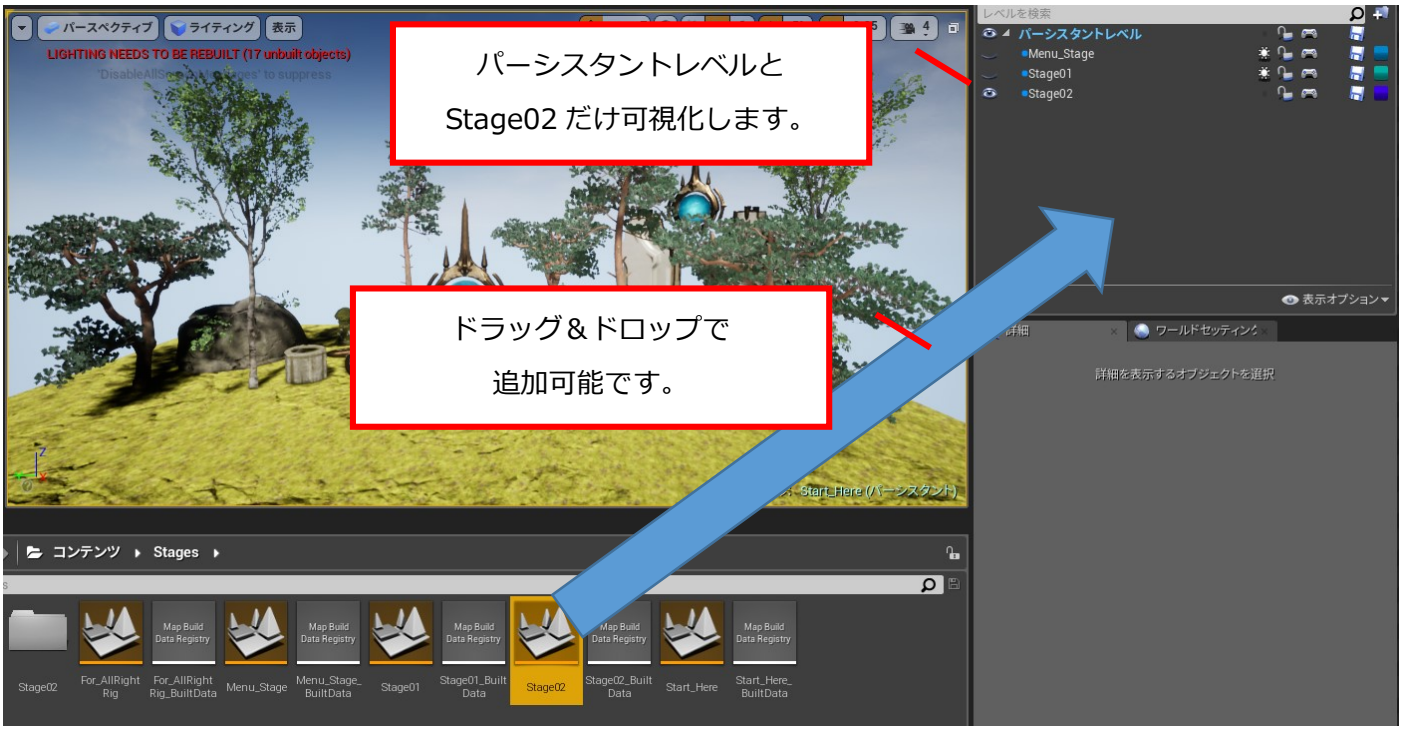
このプロジェクトで、カメラに向かって正面となるのは、以下の向きの座標軸です。

ここまで確認できたら、好みのレベルを作成してください。

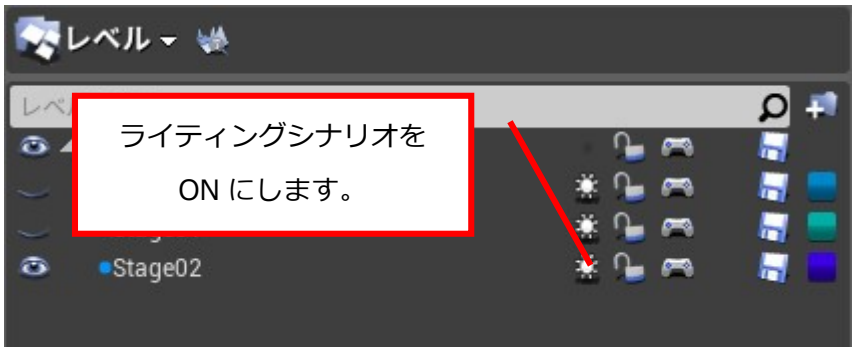
ここでは、Paragon の無料アセットを事例として作成します。

Stage02 のライティングビルドをして、保存したら、StartHere レベルを開き、Stage02 をサブレベルとして追加します。

Stage02 の地面の座標を X:0 Y:2000 Z:-70 としてください。



上記の状態で、Stage02 のライティングシナリオを ON にして、ライティングビルドを行います。



これ以降、StartHere レベルでサブレベルを編集する際は、可視化するのはどれか一つにしてください。

例えば、Stage01 と Stage02 を同時に可視化すると警告が表示されます。

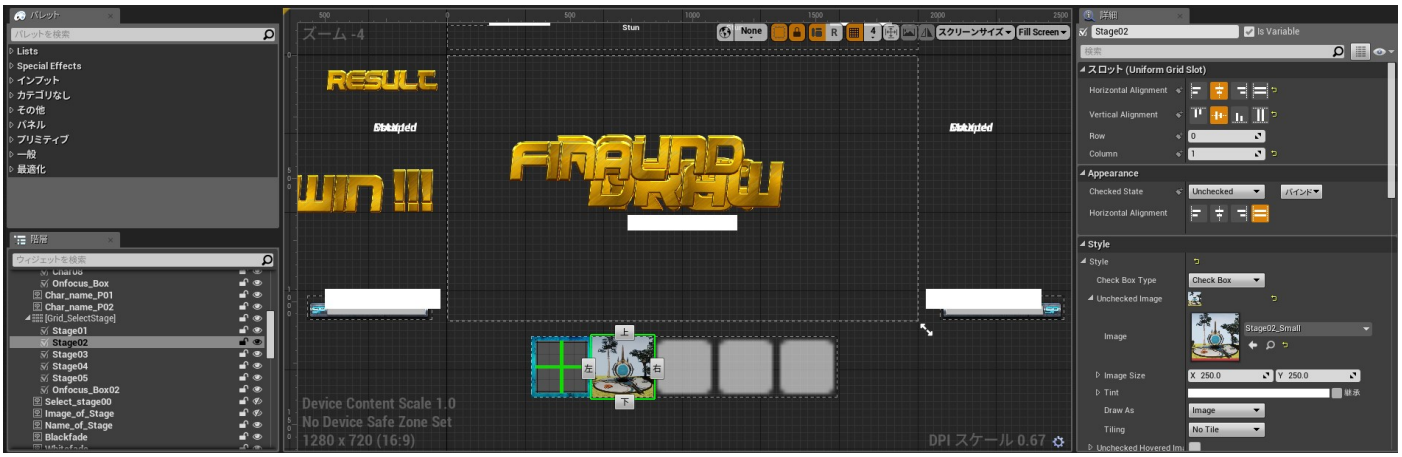
次にゲーム実行画面内のステージセレクト画面で表示するステージ用の画像を準備します。

Stage02_Small.png(250*250pixel)

Stage02_img.png（640*480Pixel）

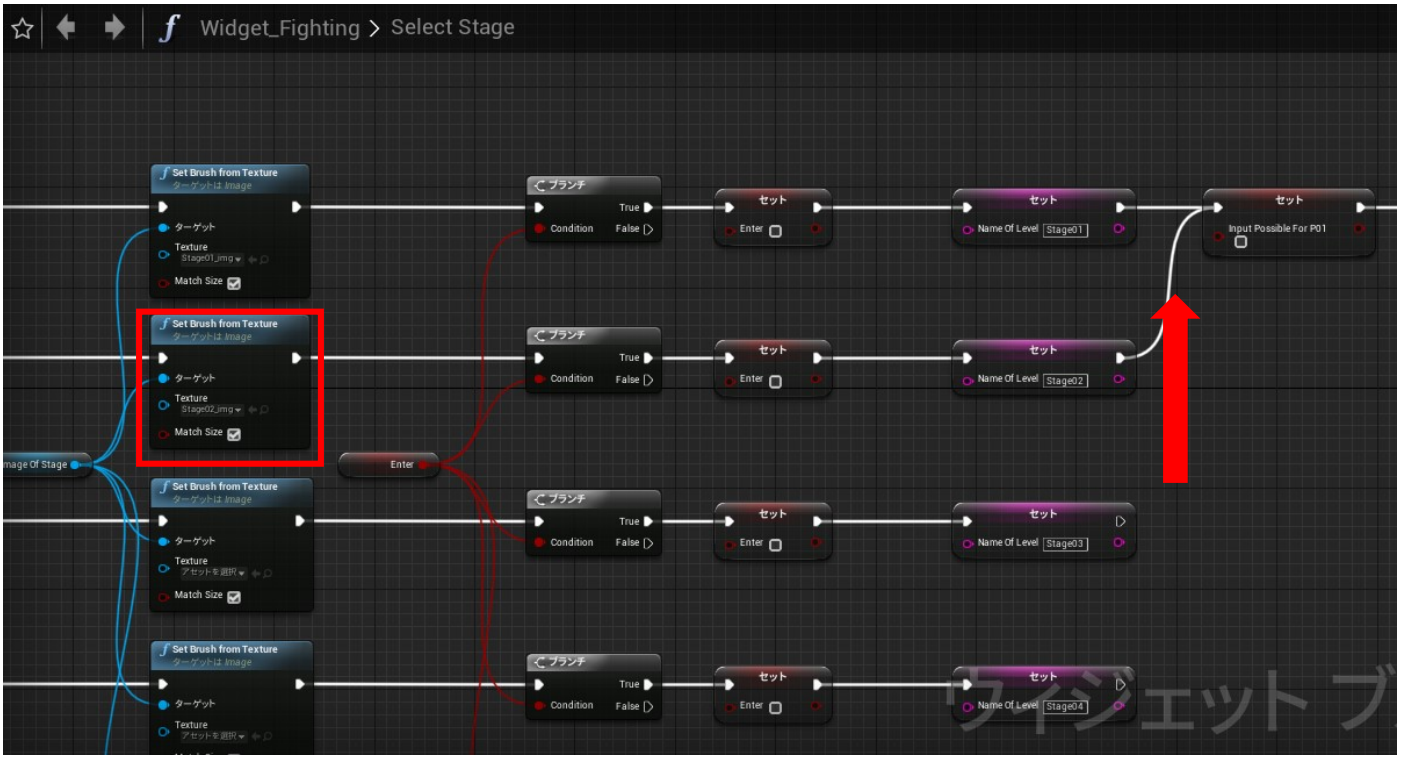
作成した画像は Widget→images フォルダへインポートしておきます。

次に Widget ブループリントを開き、デザイナー画面で以下のように、Stage02_Small.png を指定します。

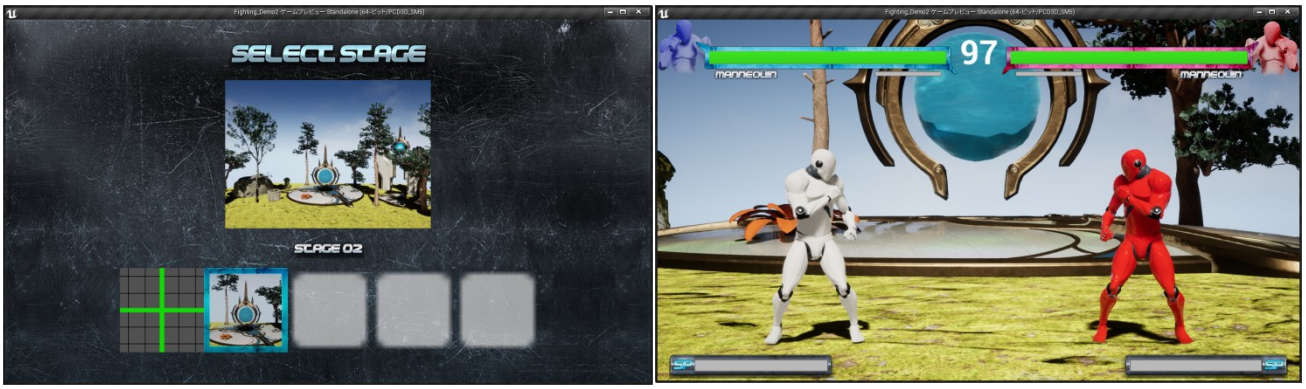


[Grid_SelectStage]にある Stage02 を選択し、Uncheckedimage に Stage02_Small.png を選択します。

次に Widget ブループリントのグラフ画面で Select Stage の関数を開き、Set Brush from Texture に Stage02_img.png の指定と選択後に処理を続けるノードの接続を行います。



これでビルドを行い、セレクトステージ画面とバトル画面で以下のように反映されていれば OK です。



5．キャラクターの追加方法

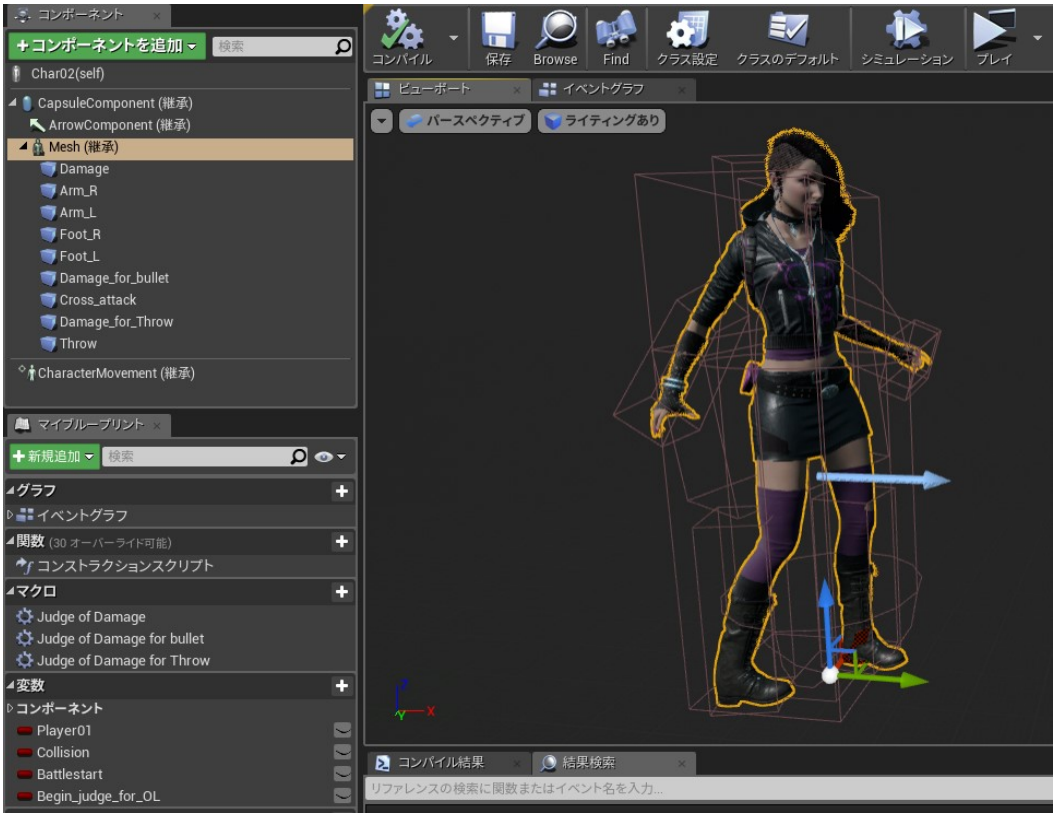
キャラクターを追加する場合は、Characters フォルダに Char02 フォルダを追加します。

ここでは、Paragon アセットの Phase を事例として説明します。

キャラクター BP のファイル名称を BP_Char02、アニメーション BP を Char02_AnimBP とします。

■キャラクターブループリントの設定

まずは、キャラクター BP の BP_Char02 を開き、Mesh や Capsule component、攻撃、ダメージを判定する Box Collision の設定を行います。



・ Capsule component

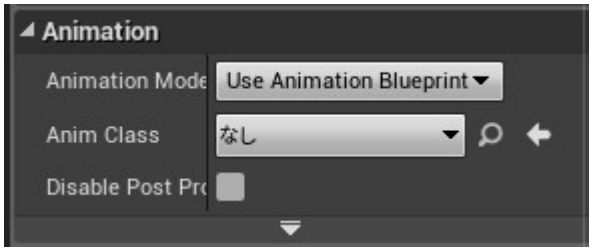
Shape の値の以下のように設定します。

Collision と Tag を以下のように設定します。

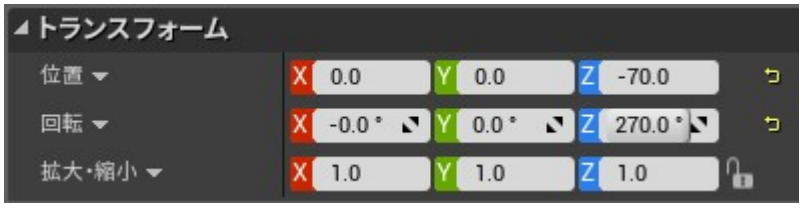
・ Mesh

Animation の Anim Class は”なし”にしてください。

キャラクターセレクト時に Animation Blueprint は必要ないため、バトル前に手動で呼び出します。



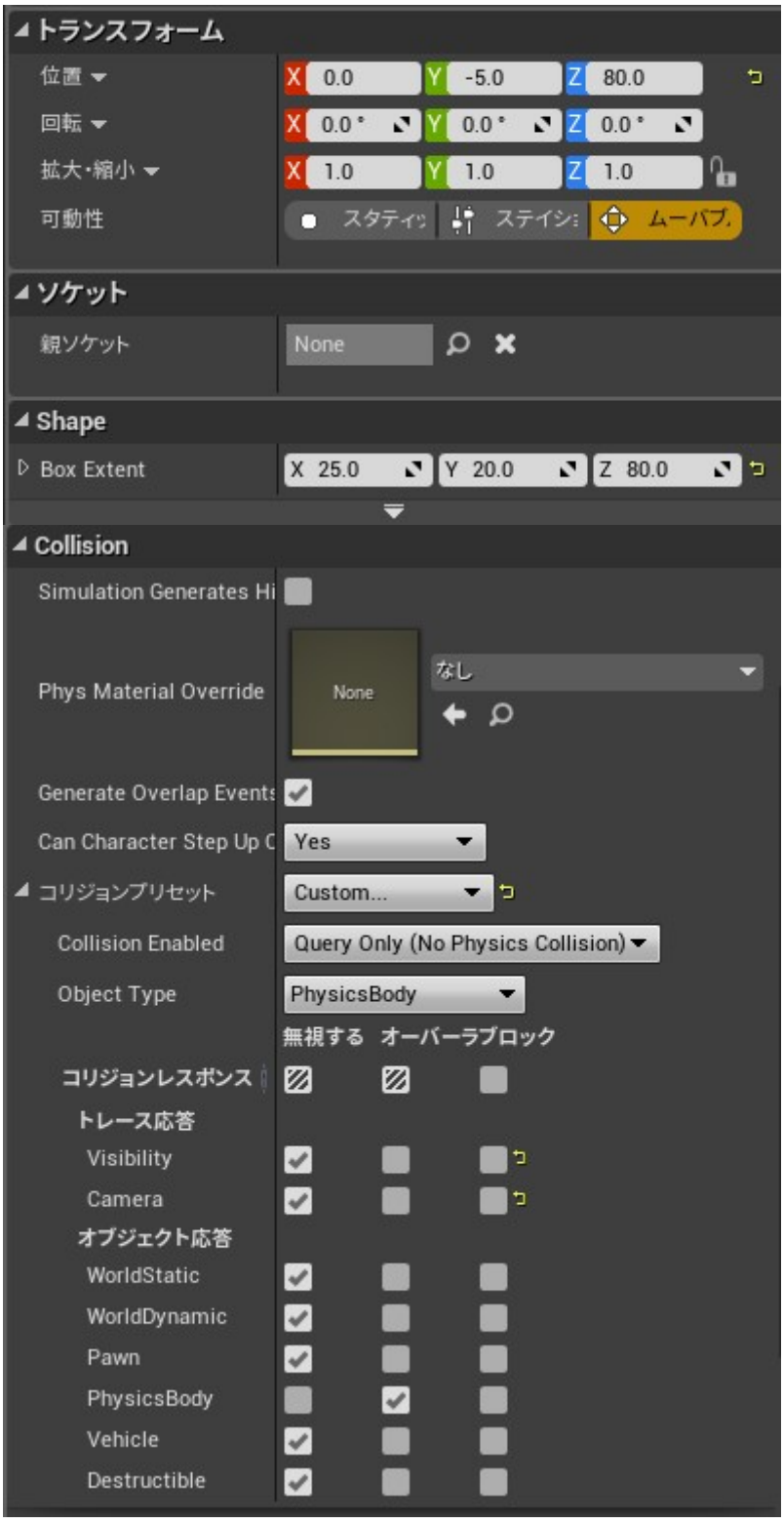
Mesh の位置と回転を以下のように設定します。



・Damge Box Collision （Mesh のサブコンポーネントとして追加します。）

位置と回転、Shape、コリジョンを以下のように設定します。

位置と Shape についてはキャラクターの体格に合わせて設定しますので、変更しても構いませんが高さをあまり低くしてしまうと、他のキャラの上段攻撃が当たらなくなるので注意してください。



・他の Box Collision について

Arm_R/Arm_L/Foot_R/Foot_L/Damage_for_bullet/Cross_attack/Damage_for_Throw/Throw/Damage_lower/Arm_R_Parry_AT/Arm_L_Parry_AT/Foot_R_Parry_AT/Foot_L_Parry_AT/Arm_R_Parry_DM/Arm_L_Parry_DM/Foot_R_Parry_DM/Foot_L_Parry_DM については Char01 の設定値を参考に位置と回転、Shape、親ソケット、コリジョンを設定します。

各 Box Collision の名称について注意してください。

また、Parry_AT や Parry_DM のコリジョンボックスはオブジェクトタイプを新規に設定しています。

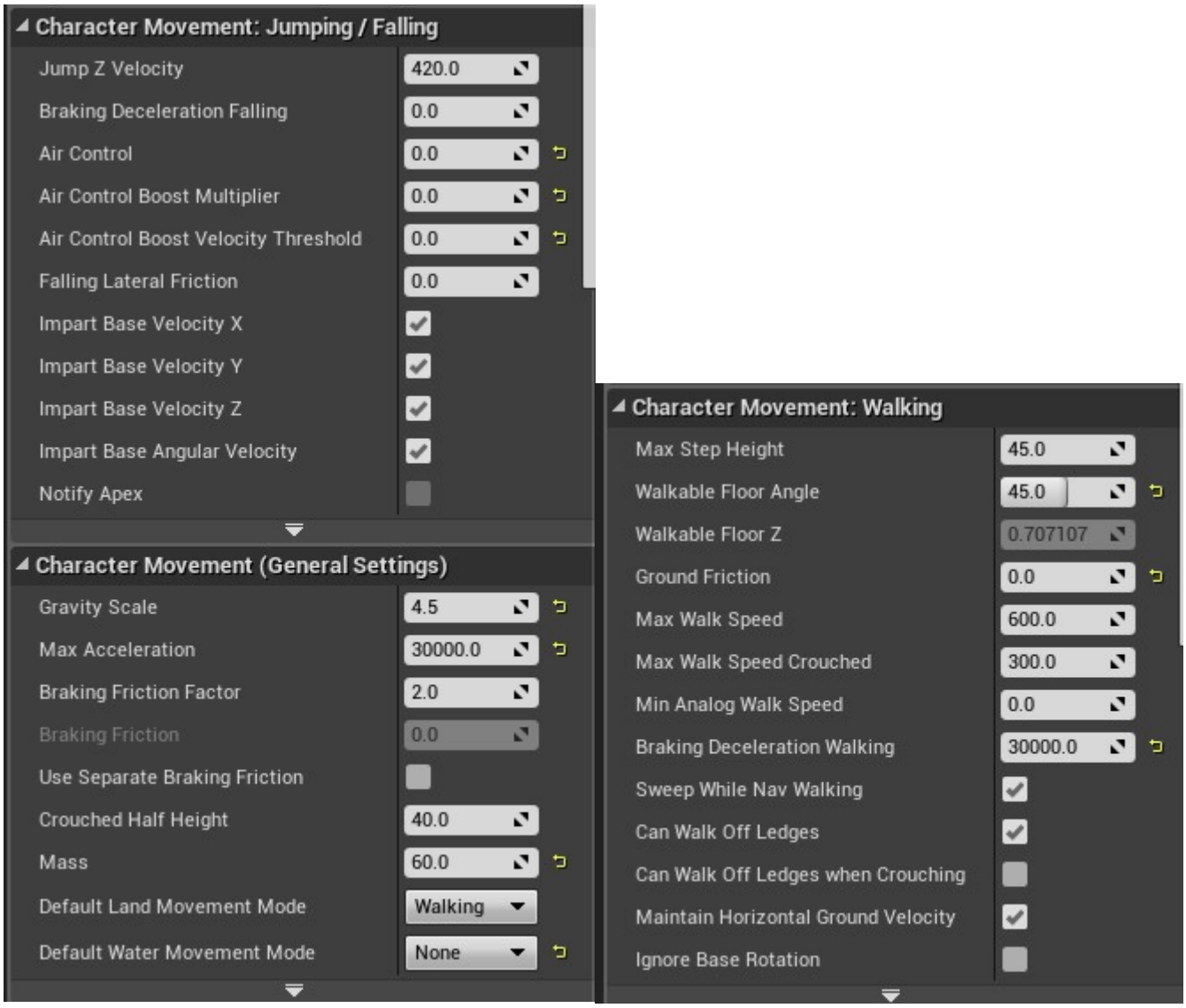
プロジェクト設定のエンジン-コリジョンの画面で新規のオブジェクトチャンネルを以下のように設定しましょう。

Parry_AT や Parry_DM の名称のコリジョンボックスの設定は以下のとおりです。

これからイベントグラフに Char01 の Blueprint をコピーしますので、間違えていると正しく動きません。

・CharacterMovement について

以下のように値を設定します。

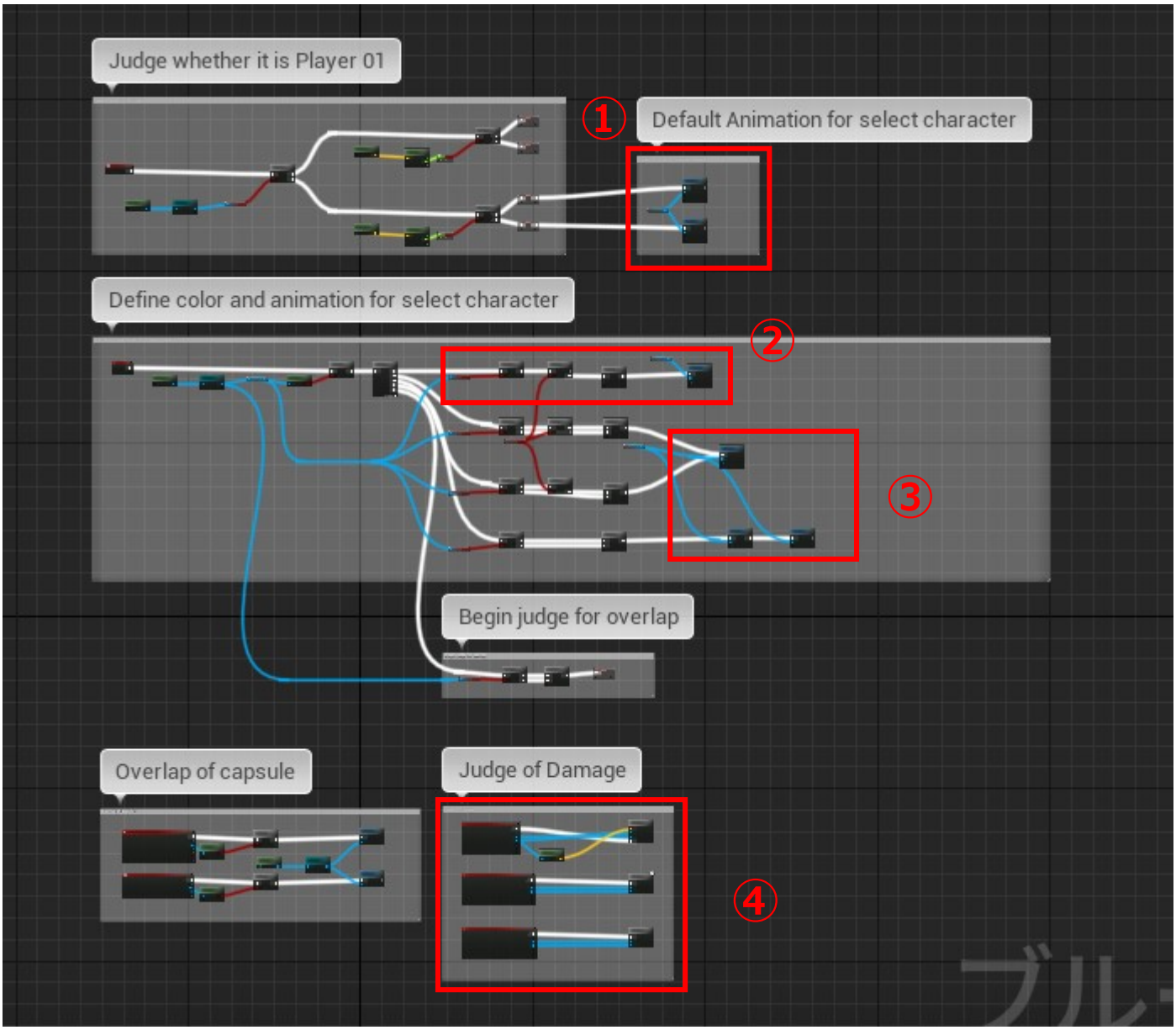


・イベントグラフについて

まずは変数を以下のように作成します。

次に Char01 のイベントグラフの Blueprint 全体を選択してコピー&ペーストします。

ただし④赤枠内はマクロがあり、マクロはコピーされませんので、Char01 のイベントグラフでマクロを展開したノードをコピー&ペーストして折りたたむかマクロ化してください。



次にキャラクタ個別で設定必要な箇所を説明します。

上図の赤枠①：キャラクターセレクト画面のキャラ選択前のアニメーションシーケンスを選択します。
上下 2 つあるのは Player1 と 2 の場合です。

上図の赤枠②：Player2 のキャラ選択時に Player1 と同キャラクターが選択された場合の処理です。
同キャラクターメッシュ内のマテリアルの色を変更するか、メッシュ自体を変更します。
メッシュ自体を変更した場合は、アニメーションシーケンスとアニメーションモンタージュをそのメッシュ専用で別途用意する必要がありますので注意してください。

上図の赤枠③：キャラクター選択時のアニメーションシーケンスと使用するアニメーションブループリントを設定します。

これでキャラクターブループリントの設定は完了です。

■アニメーションブループリントの設定

設定にはアニメーションシーケンスとモンタージュが必要になるので、以下に一覧を記載します。

準備が必要なアニメーションシーケンス一覧	
立ち待機状態	立っている状態で何も入力されていないときのアニメーション。Char01 の Idle が該当。
前歩き	前進するときのアニメーション。Char01 の Fwd_Walk が該当。
後ろ歩き	後退するときのアニメーション。Char01 の Bwd_Walk が該当。
しゃがみ待機状態	しゃがんでいる状態のアニメーション。Char01 の Crouch_Idle が該当。
しゃがみ→立ち移行	しゃがんでいる上から立つときのアニメーション。Char01 の End_Crouch が該当。
ジャンプ開始	立ち状態からジャンプを始めるときのアニメーション。Char01 の Start_Jump が該当。
空中待機状態	ジャンプが完了したときのアニメーション。Char01 の Loop_Jump が該当。
着地	ジャンプから着地するときのアニメーション。Char01 の End_Jump が該当。
前ステップ	前ステップするときのアニメーション。Char01 の Fwd_Dash が該当。
バックステップ	バックステップするときのアニメーション。Char01 の Bwd_Step が該当。
立ちパンチ弱	立ちパンチ弱をするときのアニメーション。Char01 の ST_Punch が該当。
立ちパンチ強	立ちパンチ強をするときのアニメーション。Char01 の ST_L_Punch が該当。
立ちキック弱	立ちキック弱をするときのアニメーション。Char01 の ST_Kick が該当。
立ちキック強	立ちキック強をするときのアニメーション。Char01 の ST_L_Kick が該当。
立ちアッパー	チェーンコンボで相手を上に打ち上げるアニメーション。Char01 の ST_Upper が該当。
ジャンプパンチ弱	ジャンプパンチ弱をするときのアニメーション。Char01 の Air_Punch が該当。
ジャンプパンチ強	ジャンプパンチ強をするときのアニメーション。Char01 の Air_L_Punch が該当。
ジャンプキック弱	ジャンプキック弱をするときのアニメーション。Char01 の Air_Kick が該当。
ジャンプキック強	ジャンプキック強をするときのアニメーション。Char01 の Air_L_Kick が該当。
しゃがみパンチ弱	しゃがみパンチ弱をするときのアニメーション。Char01 の CR_Punch が該当。
しゃがみパンチ強	しゃがみパンチ強をするときのアニメーション。Char01 の CR_L_Punch が該当。
しゃがみキック弱	しゃがみキック弱をするときのアニメーション。Char01 の CR_Kick が該当。
しゃがみキック強	しゃがみキック強をするときのアニメーション。Char01 の CR_L_Kick が該当。
立ちガード	立ちガードするときのアニメーション。Char01 の Guard が該当。
しゃがみガード	しゃがみガードするときのアニメーション。Char01 の C_Guard が該当。
立ち上段やられ小	立ち状態の上段やられ小のアニメーション。Char01 の Damaged_Top が該当。
立ち上段やられ大	立ち状態の上段やられ大のアニメーション。Char01 の Damaged_Top_L が該当。
立ち上段やられ中	立ち状態の上段やられ中のアニメーション。Char01 の Damaged_Mid が該当。
しゃがみやられ	しゃがみやられするときのアニメーション。Char01 の C_Damaged が該当。
空中やられ小	空中のやられ小のアニメーション。Char01 の Damaged_inAir が該当。
空中やられ大	空中のやられ大のアニメーション。Char01 の Damaged_inAir_L が該当。
空中やられ落下	空中でダメージを受けて落下しているときのアニメーション。Char01 の Falling が該当
カウンターやられ	カウンターでやられたときのアニメーション。Char01 の Counter が該当。
KO やられ	地上で KO されたときのアニメーション。Char01 の KO が該当。
ダウン	ダウン属性の攻撃を受けたときのアニメーション。Char01 の Down が該当。
起き上がり通常	起き上がるときのアニメーション。Char01 の Get_up01 が該当。
起き上がりスタン	スタン状態から起き上がるときのアニメーション。Char01 の Get_up02 が該当。

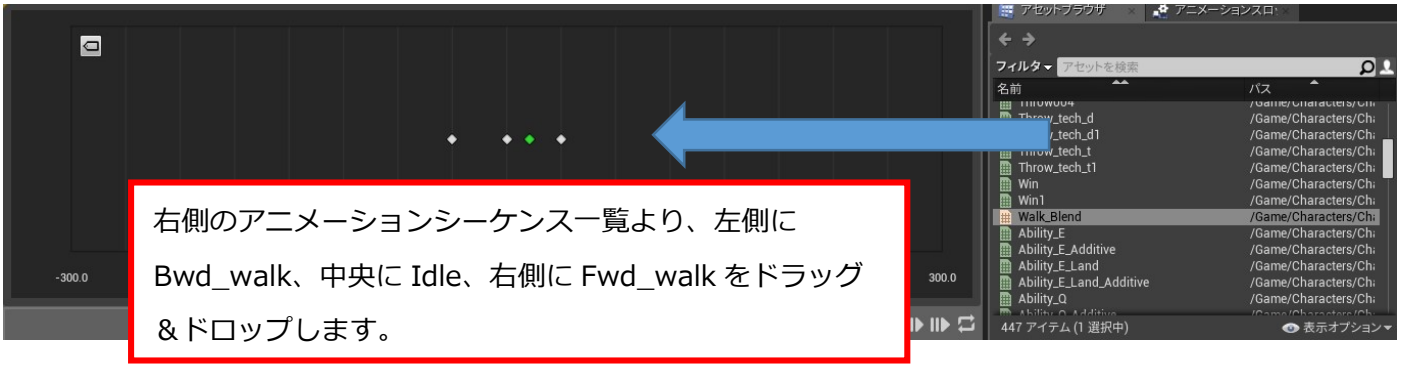
スタン	スタン状態のアニメーション。Char01 の Stun が該当。
投げ	相手を掴むアニメーション。Char01 の Throw001 が該当。
投げ成功	相手を掴んで投げるアニメーション。Char01 の Throw002 が該当。
投げやられ	投げられるアニメーション。Char01 の Damaged_Throw が該当。
投げ抜け攻め側	投げ抜けされた場合の攻撃側のアニメーション。Char01 の Throw_tech_A が該当。
投げ抜け守備側	投げ抜けした場合のアニメーション。Char01 の Throw_tech_D が該当。
必殺技（FireBall）	FireBall のアニメーション。Char01 の FireBall が該当。
必殺技（Flash Straight）	Flash_Straight 弱のアニメーション。Char01 の FS_S が該当。
必殺技（Flash Straight）	Flash_Straight 強のアニメーション。Char01 の FS_L が該当。
必殺技（Flash Straight）	空中で使用できる Flash_Straight のアニメーション。Char01 の FS_inAir が該当。
必殺技（Flash Straight）上	Flash_Straight_上_弱のアニメーション。Char01 の FLtoA_S が該当。
必殺技（Flash Straight）上	Flash_Straight_上_強のアニメーション。Char01 の FLtoA_L が該当。
必殺技（IceBall）	IceBall のアニメーション。Char01 の SP が該当。
ガード中の反撃	ガード中の反撃のアニメーション。Char01 の G-Reversal が該当。
立ち状態のスウェー	立ち状態のスウェーのアニメーション。Char01 の ST_SW が該当。
しゃがみスウェー	しゃがみ状態のスウェーのアニメーション。Char01 の CR_SW が該当。
立ち状態パリング	立ち状態のスウェーのアニメーション。Char01 の ST_PR が該当。
しゃがみパリング	しゃがみ状態のスウェーのアニメーション。Char01 の CR_PR が該当。
ラウンド勝利ポーズ	ラウンド勝利時のアニメーション。Char01 の Win が該当。
ラウンド敗者ポーズ	タイムアウト時のラウンド敗者のアニメーション。Char01 の Draw が該当。
ゲーム勝利ポーズ	ゲーム勝利時ののアニメーション。Char01 の GameSet が該当。
チェンジサイド	キャラクターが向きを変えるときのアニメーション。Char01 の Side_Change が該当。
交代時の攻撃	3on3 でキャラクターが交代するときのアニメーション。 Char01 の Change と Change_end が該当。

準備が必要なアニメーションモンタージュ一覧	
ST_Attack	立ち状態の攻撃アニメーションを登録します。AnimNotify で、カウンター攻撃を受ける時間帯、攻撃判定が発生する時間帯、必殺技へのキャンセルを受け付ける時間帯、攻撃判定消失、攻撃中フラグのオフを設定します。※
CR_Attack	しゃがみ状態の攻撃アニメーションを登録します。AnimNotify は ST_Attack と同様です。
Attack_in_Air	ジャンプ状態の攻撃アニメーションを登録します。AnimNotify は攻撃判定が発生する時間帯、攻撃判定消失、攻撃中フラグのオフを設定します。
Change	交代時の攻撃アニメーションを登録します。AnimNotify は攻撃判定消失、攻撃中フラグのオフを設定します。
SM	必殺技のアニメーションを登録します。AnimNotify は ST_Attack と同様です。
Dash	前ステップやバックステップのアニメーションを登録します。AnimNotify はステップのオフとバックステップの無敵時間の解除タイミングを設定します。
Guard	立ちガードとしゃがみガードのアニメーションを登録します。AnimNotify の設定はありません。
Damage	ダメージを受けたときの各種アニメーションを登録します。AnimNotify はダメージフラグをオフするタイミングの設定やカウンターフラグのオフを設定します。
Throw	投げや投げ抜けのアニメーションを登録します。AnimNotify で、カウンター攻撃を受ける時間

	帯、攻撃判定が発生する時間帯、投げ抜けを許容する時間帯、攻撃判定消失、攻撃中フラグのオフを設定します。
Other	ダウンや起き上がり、スタン、KO、向きの変更のアニメーションを登録します。 AnimNotify で、ダウンや起き上がりフラグのオフを設定します。
Result	勝利、敗北のアニメーションを登録します。AnimNotify の設定はありません。

※Char01 のアニメーションモンタージュを Char02 のメッシュでリターゲットし、アニメーションモンタージュ内のアニメーションシーケンスを入れ替えれば、AnimNotify を再利用でき、通知のタイミングを調整するだけで済みます。

次に Walk_Blend（ブレンドスペース１D）を開き、前進と後退と待機状態を設定します。



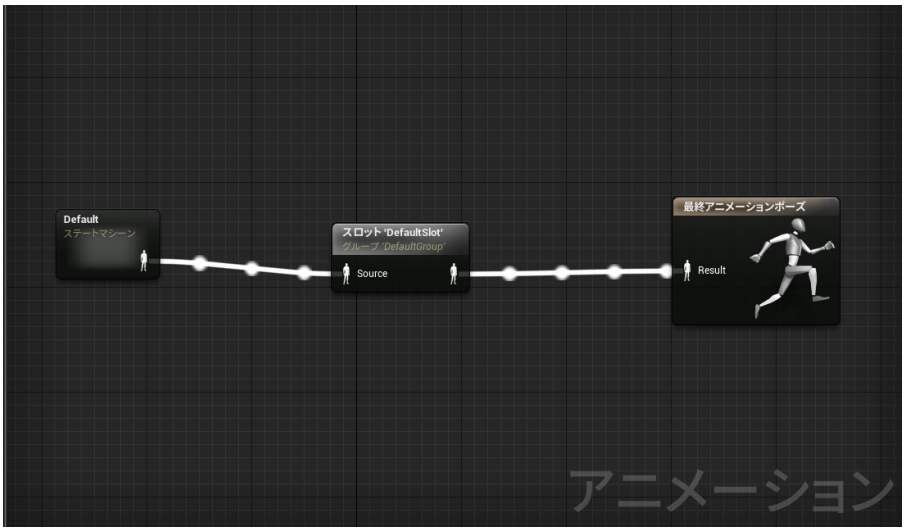
次に以下のように値を設定します。

水平軸の Name を Speed にすることが重要で他の値は参考値となります。

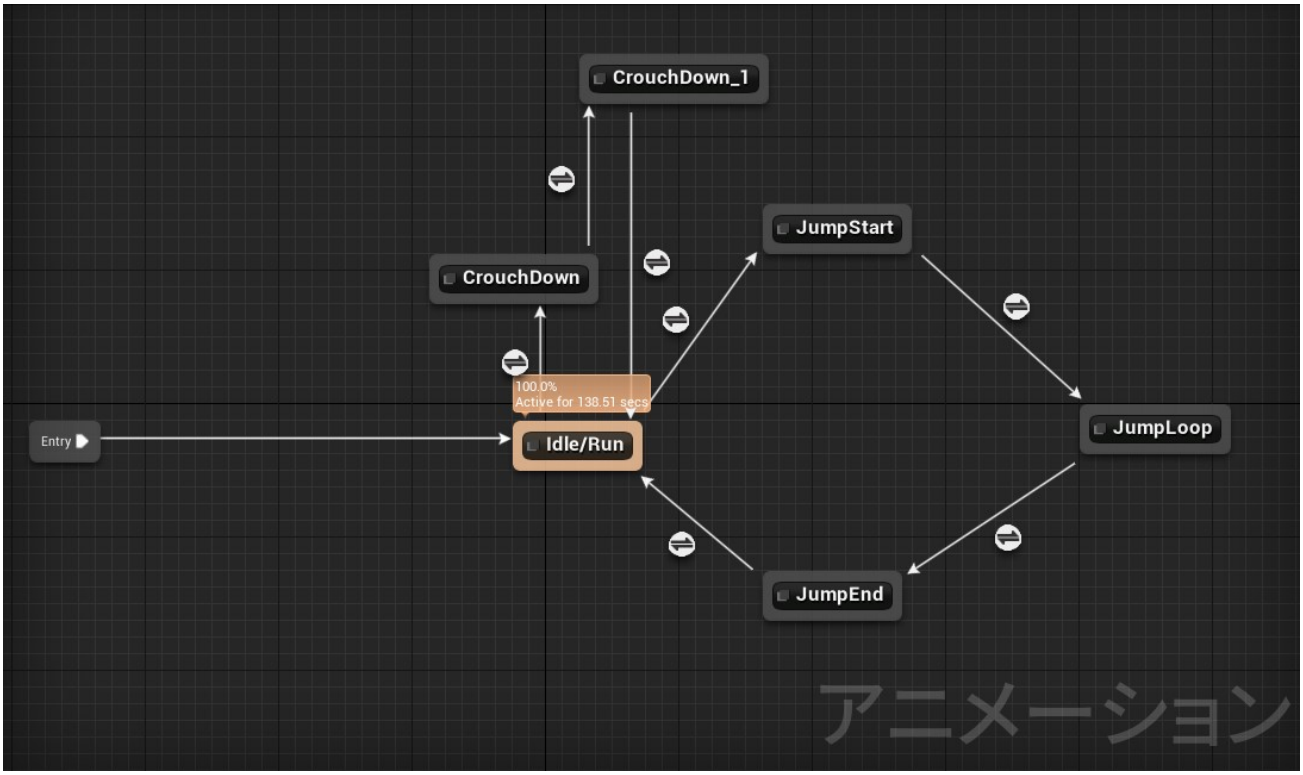
これでアニメーションシーケンスとモンタージュの準備が完了しましたので、Char02_AnimBP を開き、ステートマシーンの編集を行います。

まずは以下のように変数を設定します。

次にアニムグラフを以下のように設定します。

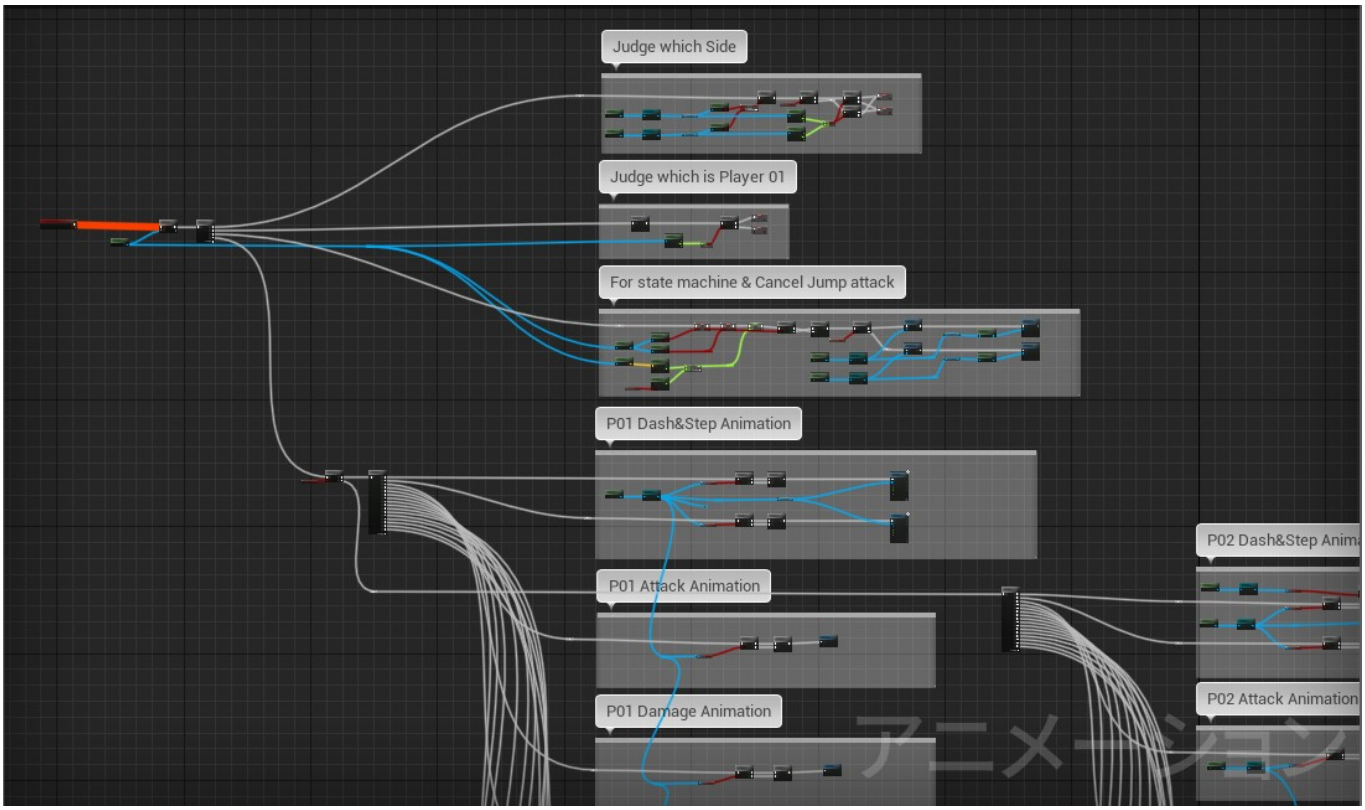


次にステートマシーンを以下のように設定します。



各項目の詳細は、Char01 のステートマシーンを参考に設定してください。

次にイベントグラフの設定ですが、Char01_AnimBP からコピー＆ペーストします。



全てコピー＆ペーストしたらビルドしてください。

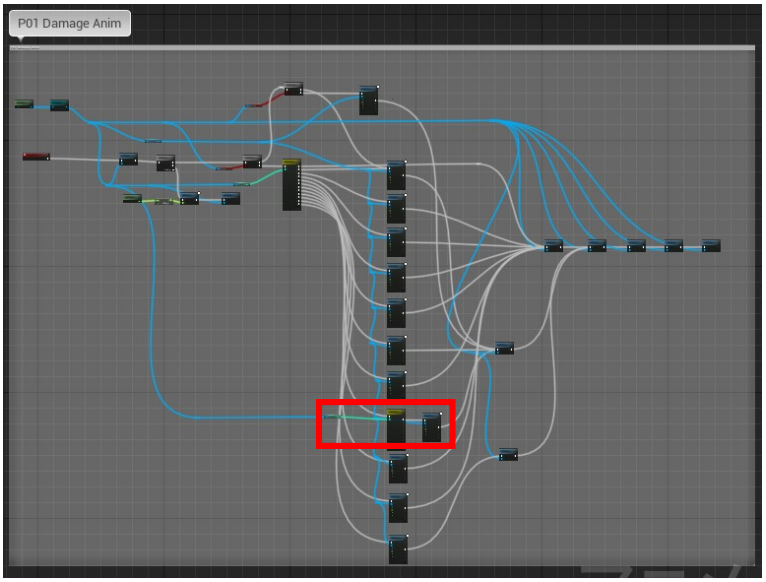
恐らく、いくつかのカスタムイベントでエラーが出ると思います。

コピー＆ペーストしただけでは有効にならない場合があるので、その場合は同じノードを作り直して、ビルドすると解消されます。

ビルドが通ったら、次に Play Montage や Montage Stop が参照しているアニメーションモンタージュを Char02 のモンタージュに設定し直します。

コピーしただけでは、Char01 のモンタージュを参照しており、ビルドが通っても動作しません。

また、ダメージを受けたときのアニメーションを処理しているノードでは投げられたときのアニメーションが各キャラ固有になる場合がありますので、以下画像赤枠部分に、個別の設定をしてください。



Play Montage や Montage Stop の参照先をすべて Char02 用に変更したら Char02_AnimBP の設定は完了です。

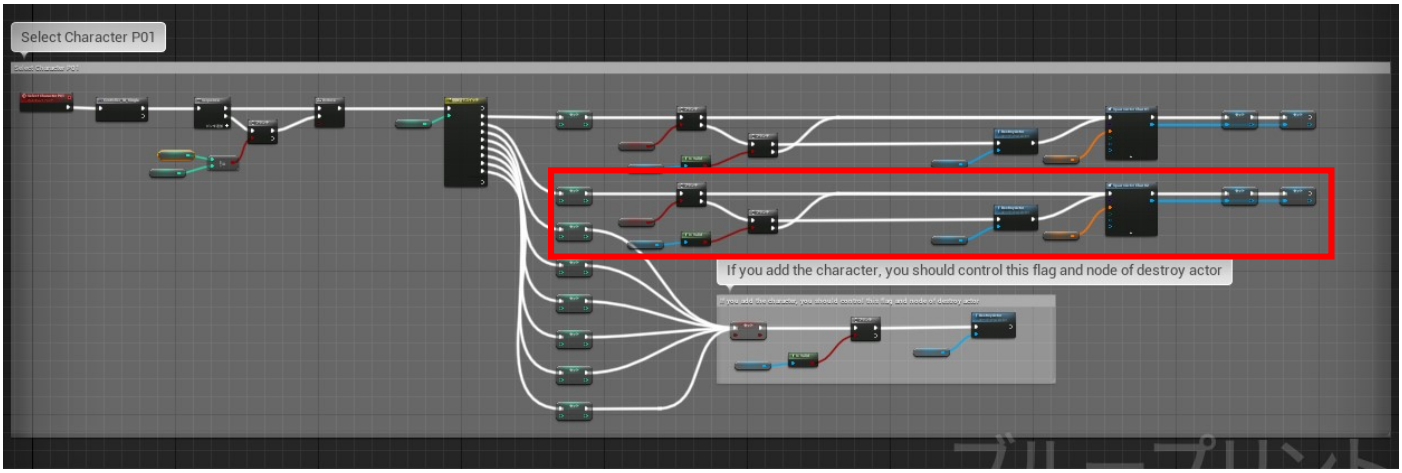
■BP_Common_Controller の設定

Char01 の設定を参考にしながら Char02 の設定を以下のように行います。

・カスタムイベント Select Character

赤枠内で Char02 の処理を追加しています。

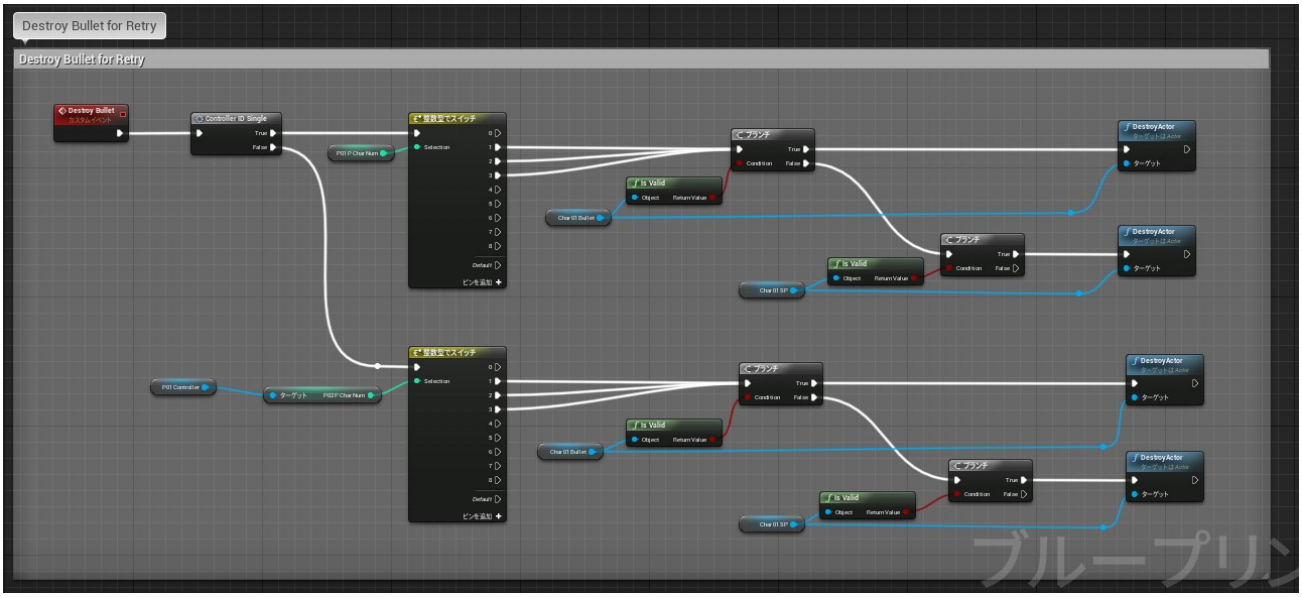
3on3 に対応させるためには、Player02 も含めた 6 箇所対応する必要があります。



・カスタムイベント　Destroy Bullet

必殺技の飛び道具を初期化する処理です。

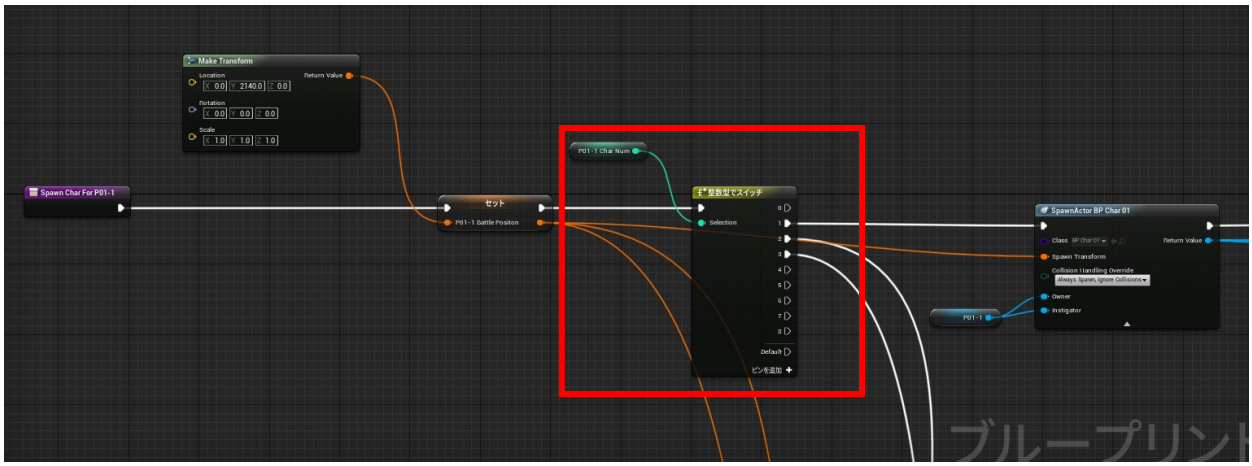
新たに設定したキャラクターの必殺技として飛び道具を設定する場合は、Char01 の設定を参考に処理を追加します。



・関数　Spawn Character

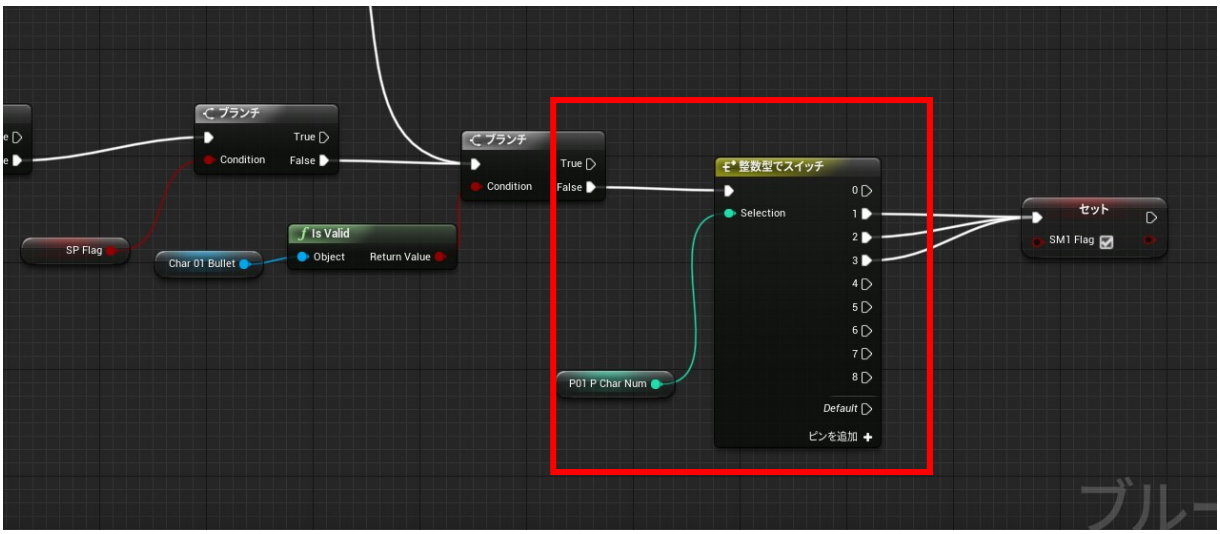
キャラクターのスポーンと初期値を設定する処理です。

3on3 に対応させるためには、Player02 も含めた 6 箇所対応する必要があります。



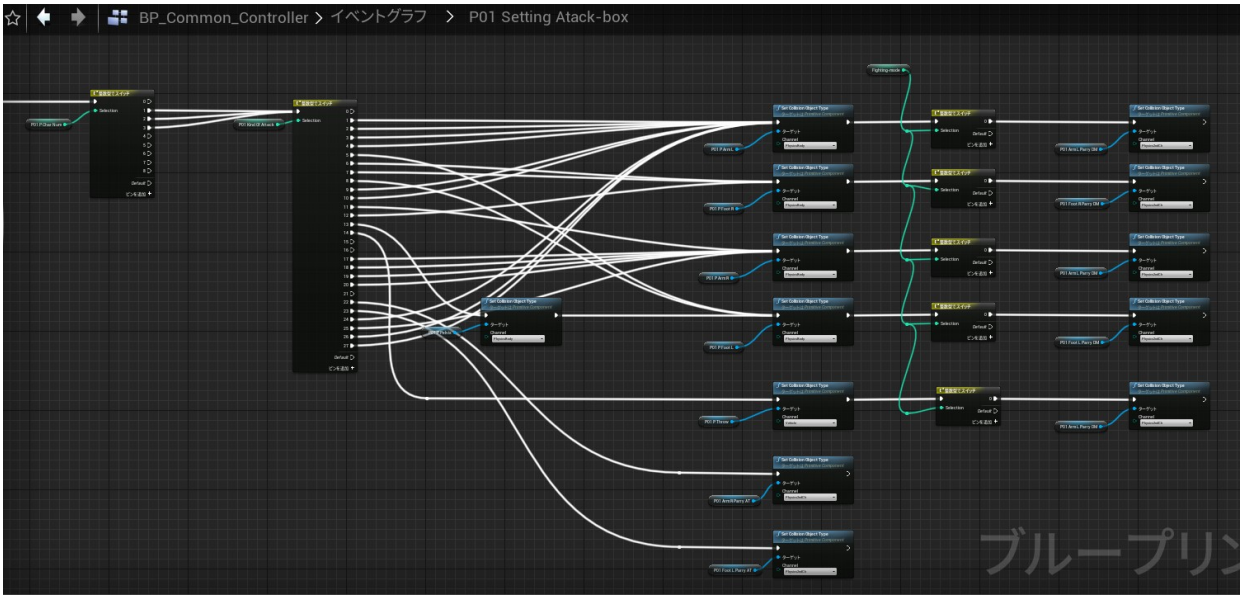
・イベングラフ　Command

236 や 214、623 のコマンドに対して、どれを有効にするか決定して、ノードを繋ぐ必要があります。



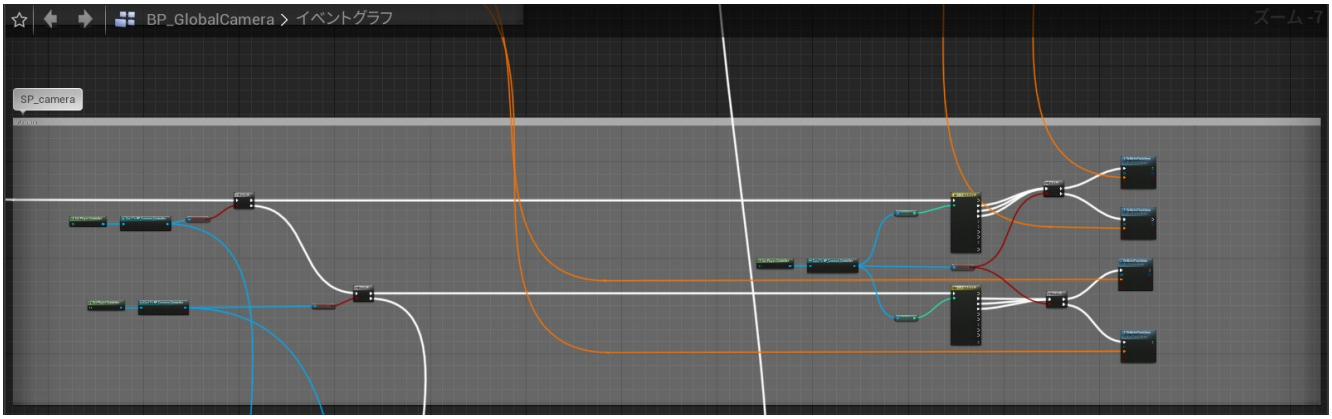
・カスタムイベント　Setting Atack-box

Kind of attacking 数値に対して、両手両足のどれを有効にするのか設定する必要があります。



■BP_GlobalCamera の設定

・SP 必殺技を使用したときのカメラの位置を設定する必要があります。

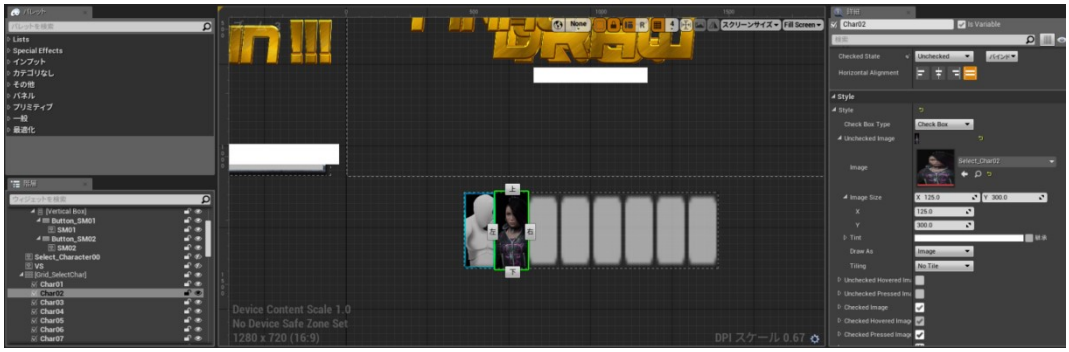


■Widget_Fighting の設定

・キャラクターセレクトやバトルで使用する画像の追加や選択されたときの設定が必要です。

まずは以下画像のようにデザイナー画面のキャラクターセレクトのグリッド背景に画像を追加します。

（125*300pixel）

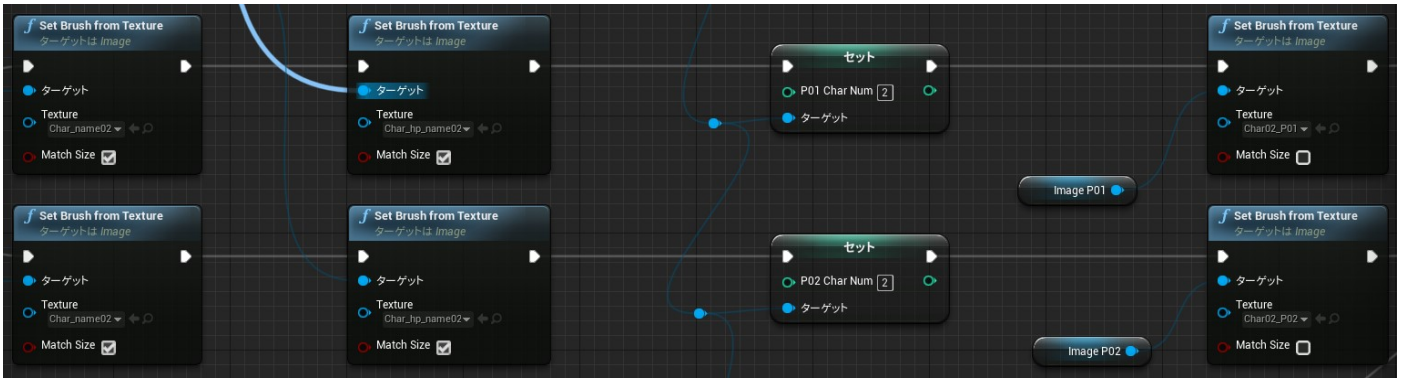


次にイベントグラフ内の Select Character 関数内で以下画像のように参照 Texture を設定します。

Char_name02（300*50pixel）：キャラクターセレクト時にキャラの下に表示される名前

Char_hp_name02（250*30pixel）：バトル時に HP バーの下に表示される名前

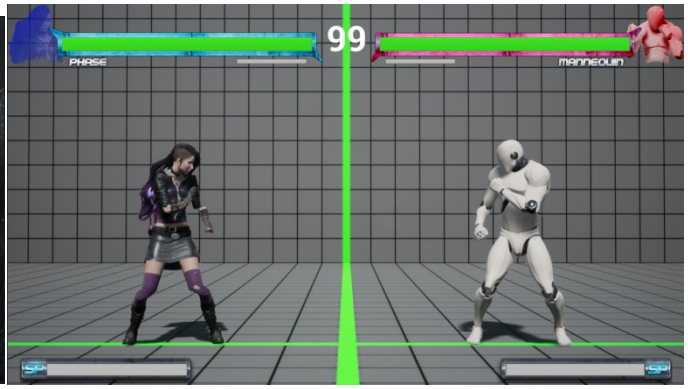
Char02_P01/Char02_P02（304*335pixel）：：バトル時に HP バーの横に表示されるキャラ画像



選択されたときの処理は、すでに設定されたノードをコピーして設定してください。

■動作確認

ビルドして実行したら、以下画面のように反映されていれば OK です。



6．デバッグ時の注意点

動作確認をする際は、必ずフレームレートを確認して行ってください。

（アウトプットログで Stat fps コマンドを入力するか Tick に繋いだ PrintString に

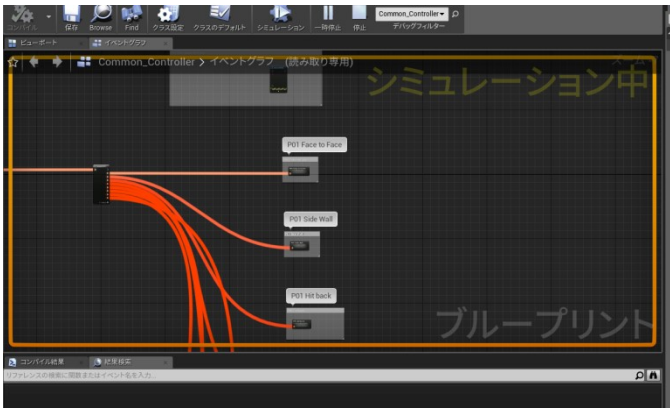
1 を WorldDeltaSeconds で割った値をセットして表示させてください。）

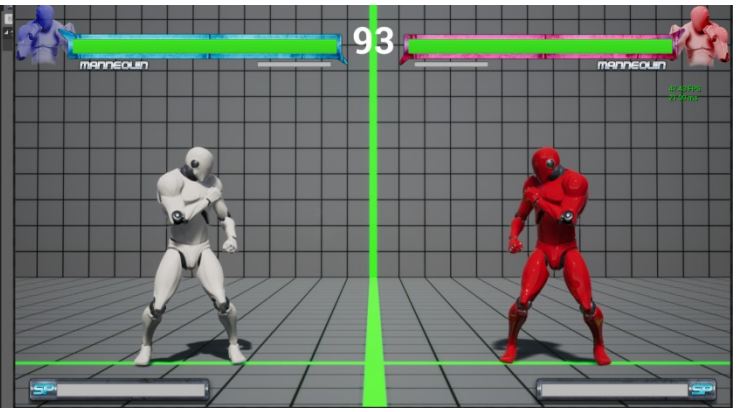
フレームレートが下がっている状態で動作すると、フレームレートが下がっている状態でしか発生しない不具合が発生することがあります。

以下画像のように Blueprint の実行を確認しながらプレイを行うとフレームレートが下がることを確認しています。

これを解消するためにはマルチディスプレイを利用して、Blueprint の実行ウインドウとプレイウインドウを別々のモニターで実行すると 60fps をキープしながら Blueprint の実行を確認できます。

ただし、Blueprint の実行確認範囲をズームアウトして広範囲にするとマルチディスプレイにしても、フレームレートが下がる場合があります。





以上

RareEncounter Ver.3.0