UnrealEngine4 で TPS 作ろう

本日は簡単に UE4 上を使って、「TPS っぽいもの」を作ってみます。
今回は alwei 氏の
https://www.slideshare.net/masahikonakamura50/5ue4
を参考にさせてもらってこの資料を作っています。今回はアセット的なものは使わないので、見た目はかなり初歩的なことになりますが、慣れたらアセット作るか買うかして豪華にしていきましょう。

今からちょっと速めのハンズオンをやっていきます。時間ないので僕の速度でやっていきます(20 分で終わる予定)ので、ついていけない人は資料を読み進めながら聞いてください。
「新規プロジェクト」タブを選択して、「ブループリント」タブを選択した状態で、ThirdPerson を選択して、「プロジェクトを作成」ボタンを押してください。

PC のスペックに不安がある方は、下段真ん中の「ハイエンド」ってアイコンを押して、「スケーラブルな 2D•3D に変更しておきましょう。」

そうすると漸くアンリアルエディタが立ち上がります。

エディタを操作

ひとまず、これがどういうものかを体験してもらいましょう。
上部メニューにある▲ボタンを押してみてください。

海外ゲームなどではおなじみの FPS ゲームのような画面になります。

操作法は洋ゲーやる人ならわかると思いますが、WASD で前後左右に移動して、マウスを上下左右にふる事によってカメラの向きが変わります。スペースキーでジャンプします。

今のところクリックでは何も始まりません

しかし画面がこの通り…

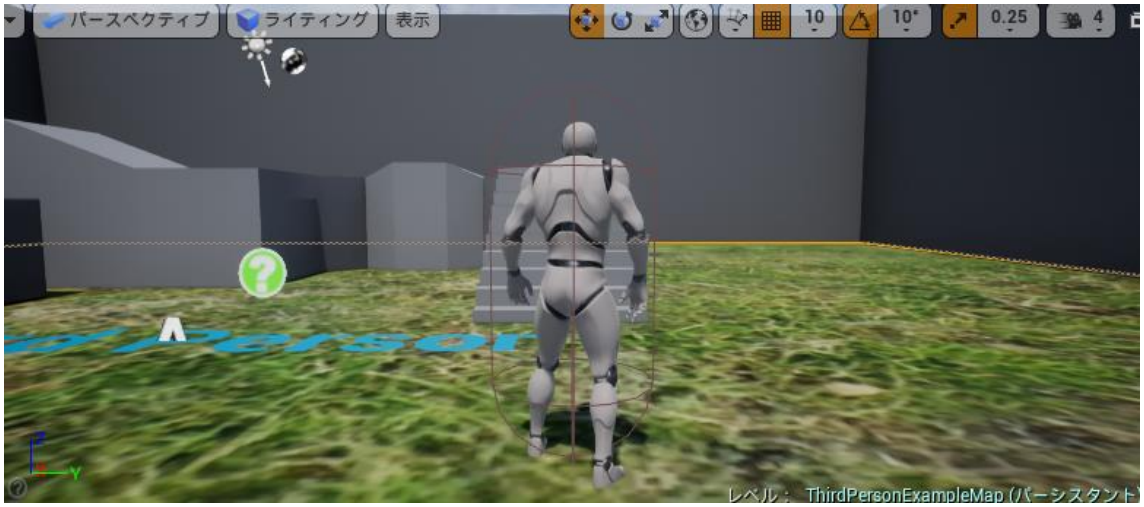
無機質ですねえ。
なお、実行をやめるには ESC キーを押します。それでもやめられない場合は、■の「停止」ボタンを押します。

材質を変更しましょう

そこで地面に土のぬくもりを与えましょう。
地面を選択して「マテリアル」を変更します。地面をクリックしてください。

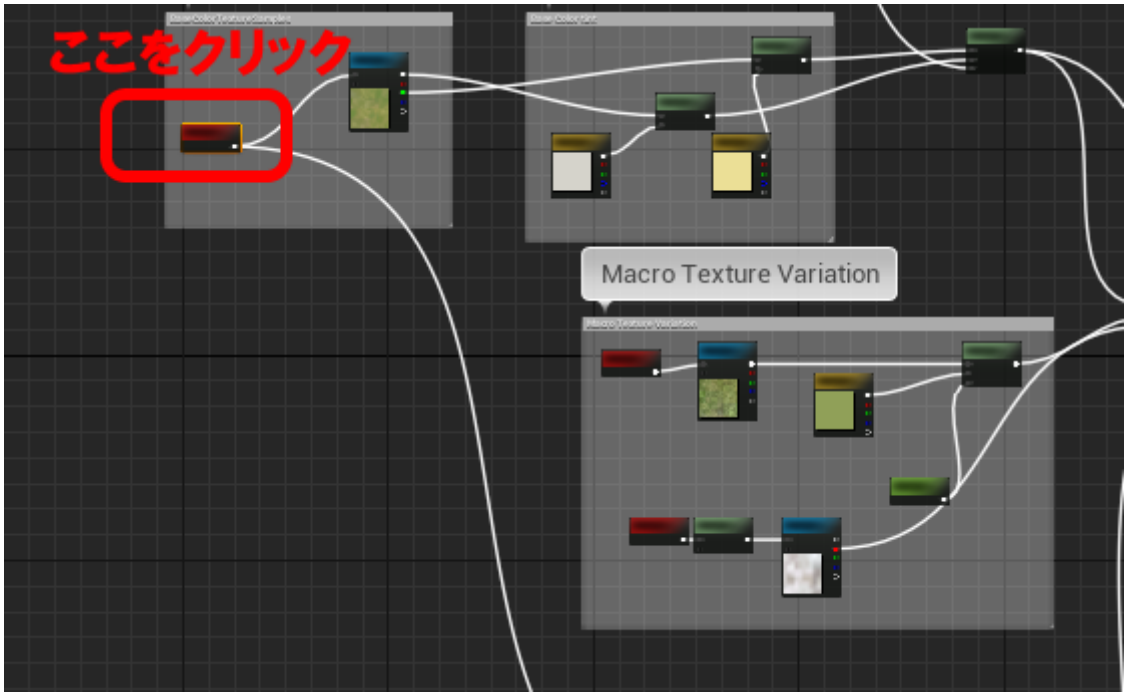
そして右側の Materials の▼を押してください。

その中から「M_Ground_Grass」を選択してください。
しばらくすると地面の材質が変わります。待ちましょう。



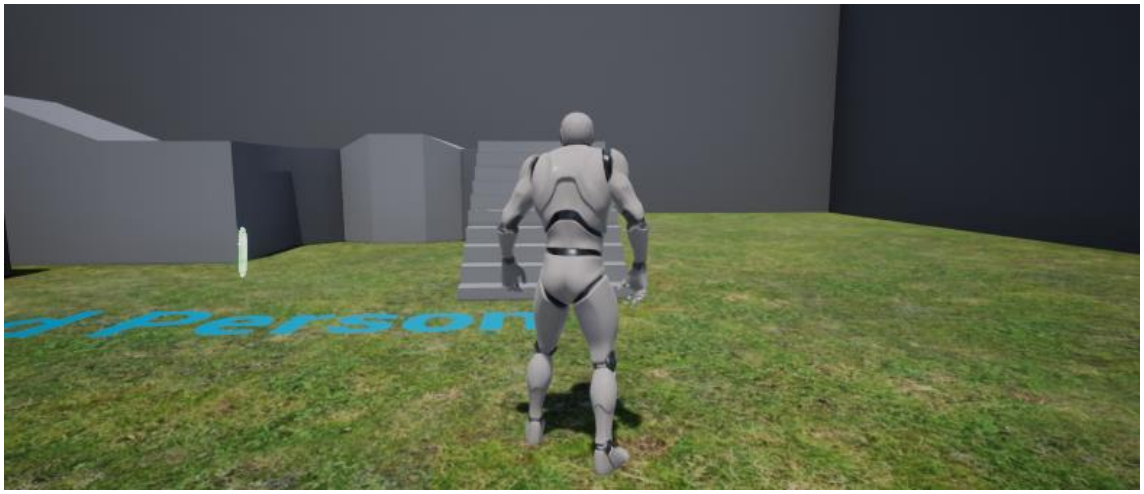
う～ん…
粗いですね。
調整しましょう。

マテリアルをダブルクリックしてください。そうすると……UE4 の醍醐味の「ノードエディタ」が立ち上がります。



左にある BaseColorTextureSamples の中にある TexCoord(0)をクリックしてください。
そうすると左側の「詳細」の部分の UTiling と VTiling が 0.5 だと思いますので、両方 5 くらいにしてください。

そうすると

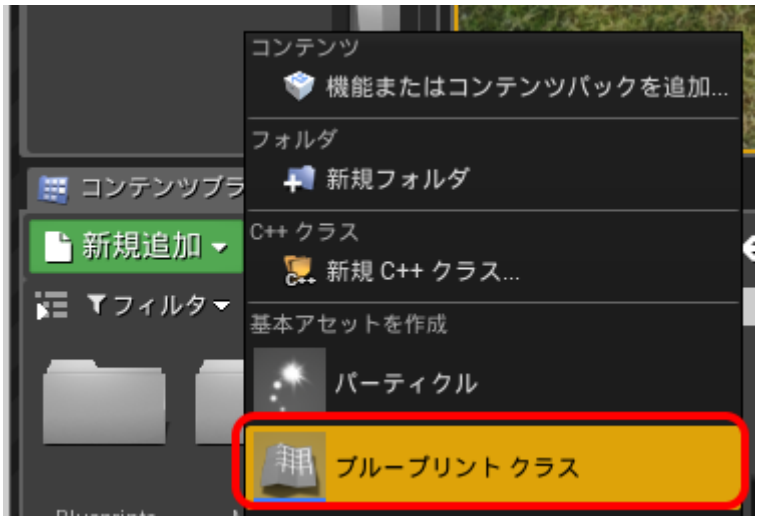


幾分マシになりましたね。

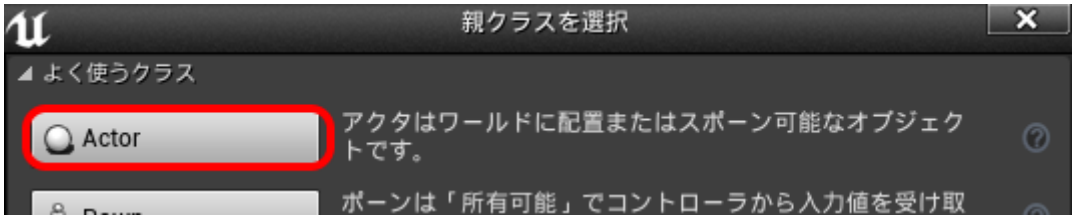
ファイアー!

しかし TPS なのに何も発射しないのは寂しい。そういうわけで弾を出せるようにしましょう。まずは弾の定義をしていきます。

新規追加→プループリントクラス



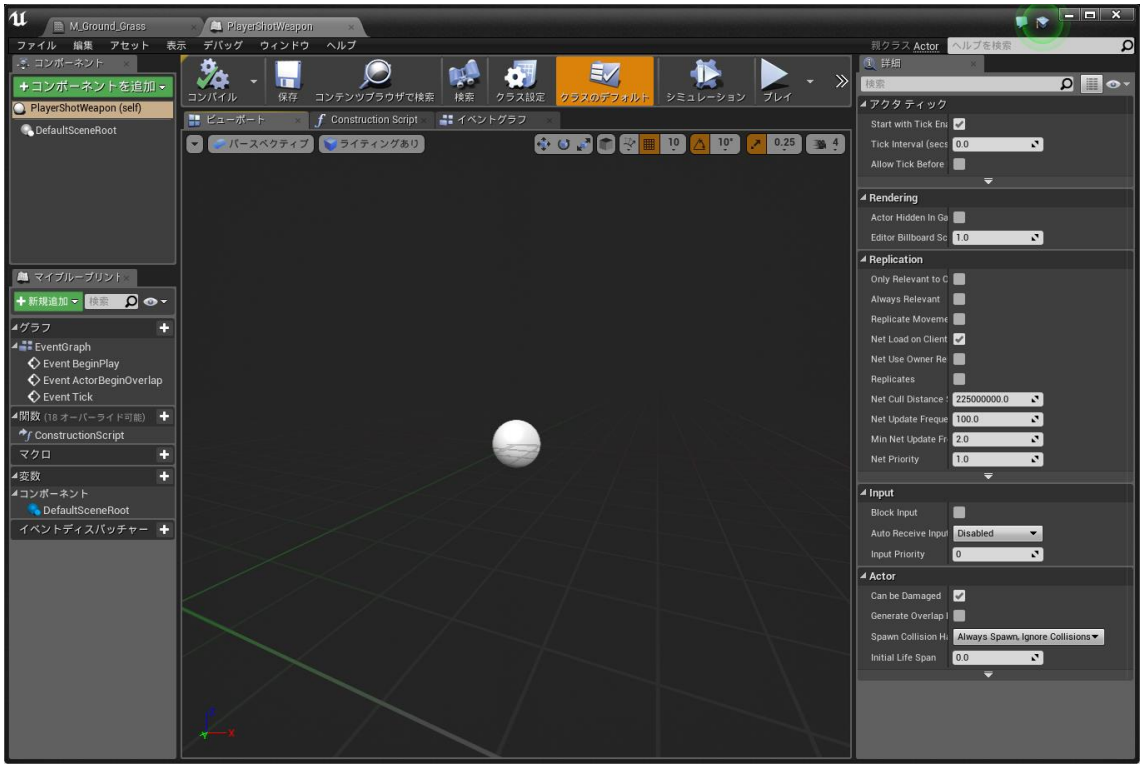
Actor を選びます。



ちなみに「スポーン」ってのは「発生」とかそんな意味なので、弾にはもってこいのクラスです。

名前は「PlayerShotWeapon」としておきましょう。

そいつをダブルクリックすると、こんな風にピンポン玉かよ!!!って画面が出てきます。

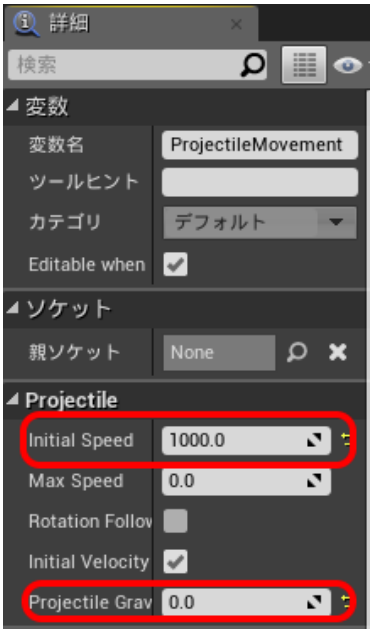


今のところ、この弾は動きませんが、これをびゅーんと飛ぶように改造してやります。

「コンポーネントを追加」をクリックして、ProjectileMovement と入力してください。
Projectile って英語は耳慣れないし、日本語の適切な訳が思いつかないのですが、イメージと

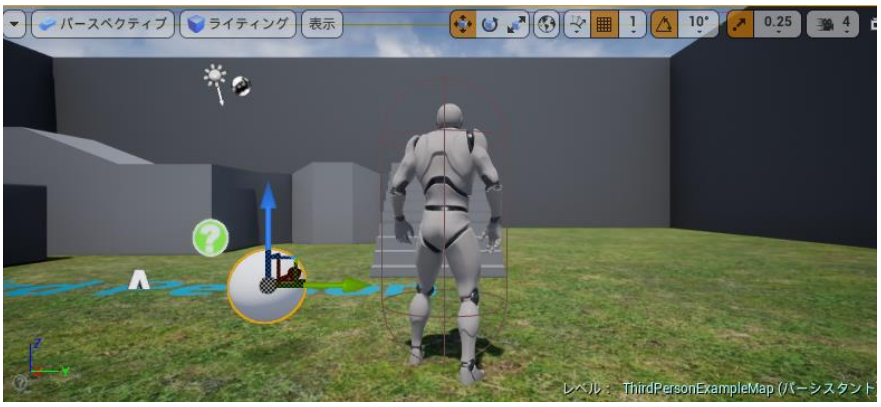
しては「びゅーん」って感じだと思ってください。矢とか弾丸とか、一定方向に進むやつを表し

ます。
でも何も設定しないと動きませんので、今度は ProjectileMovement を選択した状態で右側のパラメータを変更していきます。



InitialSpeed を 1000 くらいに。Gravity を 0 にしてください。重力の影響を受けないようにします。
ところが今のままでは本体側に配置したところで、何も表示されません。エディタに表示されているピンポン玉は、あくまでも基準の位置を示すもので、目に見えるオブジェクトじゃないわけです。
ということで、また新規コンポーネントの追加→球で、球体を追加します。大きさは最初のピンポン玉に合わせるくらいにしてください。

さて、そこまでできたら画面上に配置してみましょう。プレイヤーキャラのちょっと左くらいに置いてみましょう。



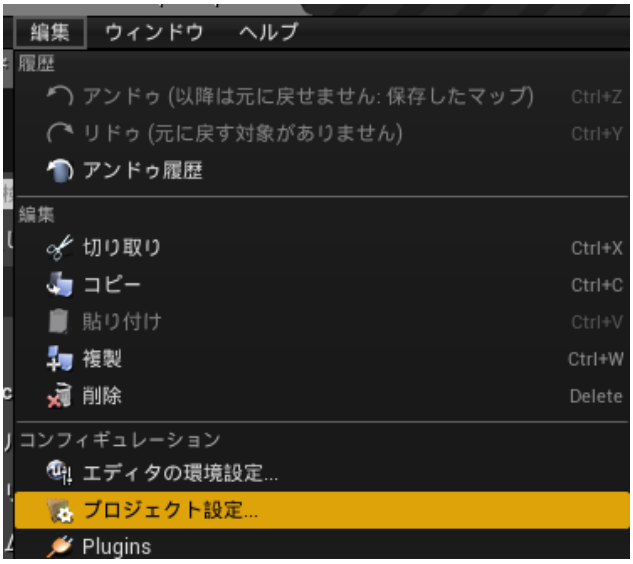
これでまた再生ボタンを押して球体が移動すれば、きちんとできています。球体が表示され

なかったり動かなかったりする場合はどこかで間違っています。

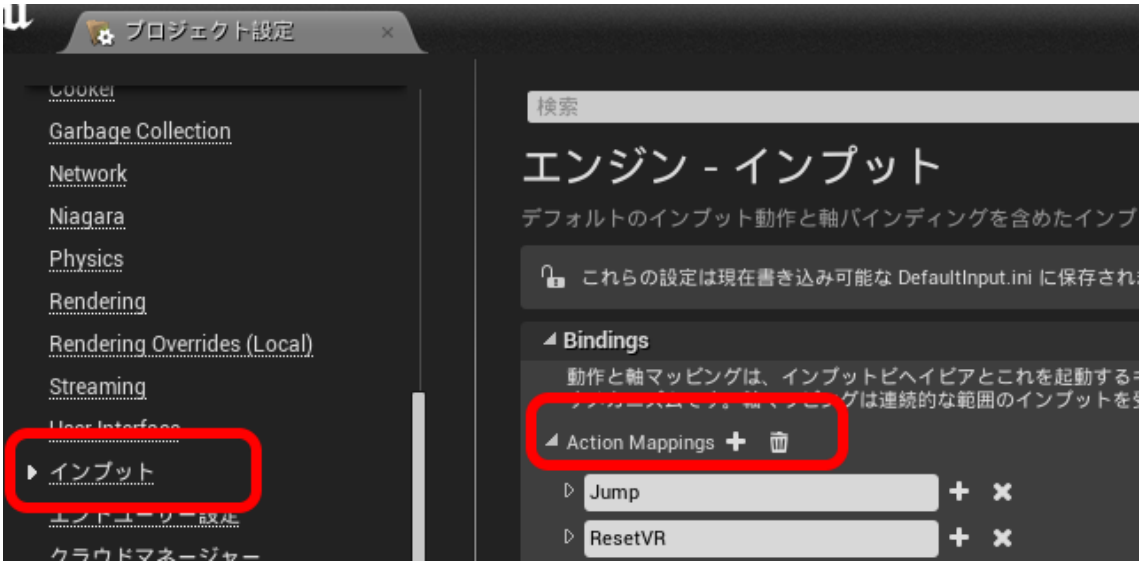
とはいえただ球体が飛ぶだけじゃ弾とは言えないね。とにかくクリックか何かをトリガーとして、それで弾が発生して飛んでいくようになってくれればいいね。

入力イベントを定義する

という事でまずは左クリックを定義しましょう。慣れてない人は戸惑うと思いますが、入力周りの設定はちょっと予想外の場所にあります(プログラマにとっては)。



編集→プロジェクト設定

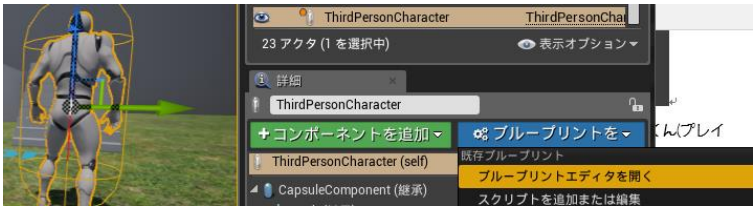


「インプット」→ActionMappings で、入力へのアクションを定義できます。

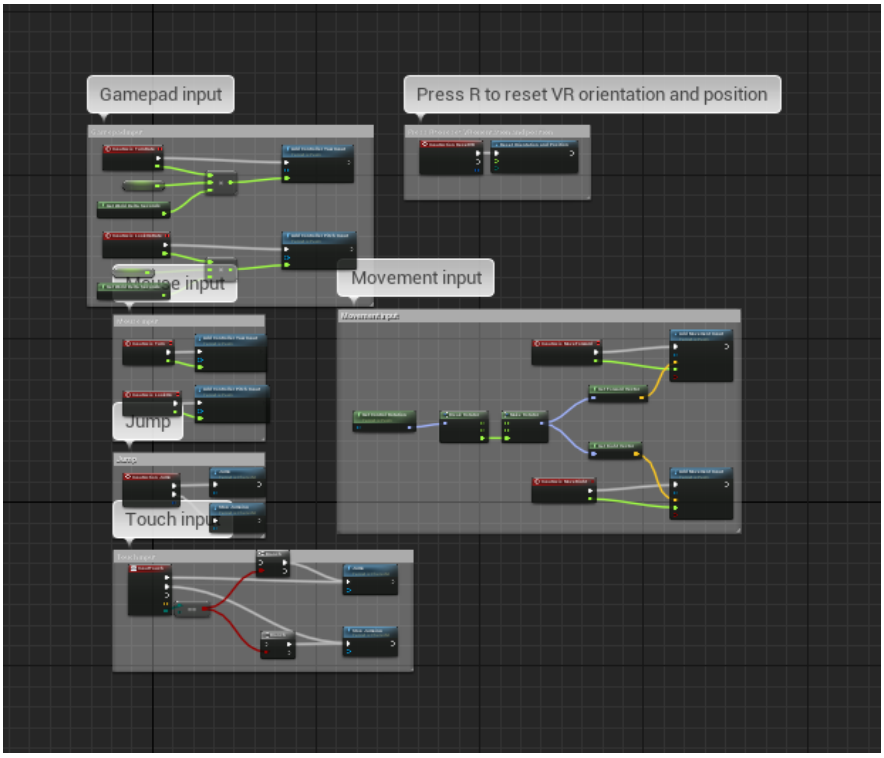
ここで ActionMapping の横の＋ボタンを押して弾の発射を定義しましょう。

名前は Shoot として、左クリックを割り当てます。

今度はそのイベントでプレイヤーから弾が飛ぶようにしたいので、ThirdPerson くん(プレイヤー)を選択した状態でブループリントを開いてください。



そうすると以下のような BP が出てきますが、最初からあるのは無視。



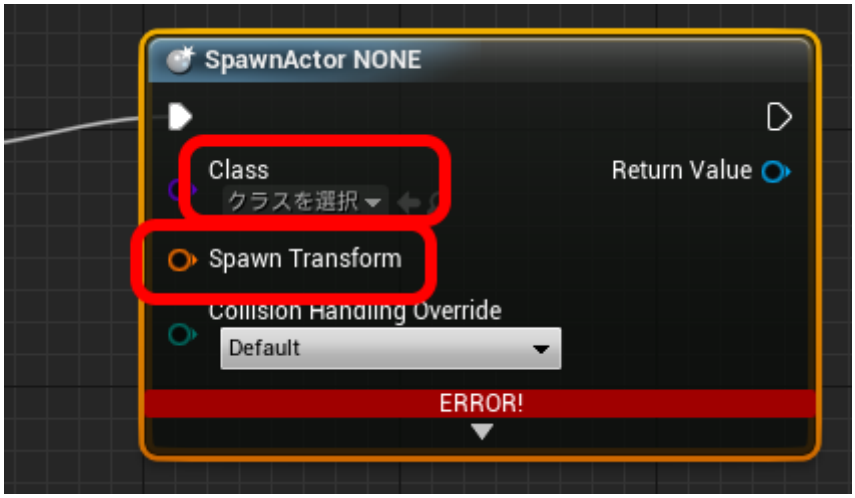
右クリックして「Shoot」と先ほどインプットイベントで登録したイベント名を入力してください。

ここから左クリックに対する反応を定義できます。勿論 Pressed が押したときです。

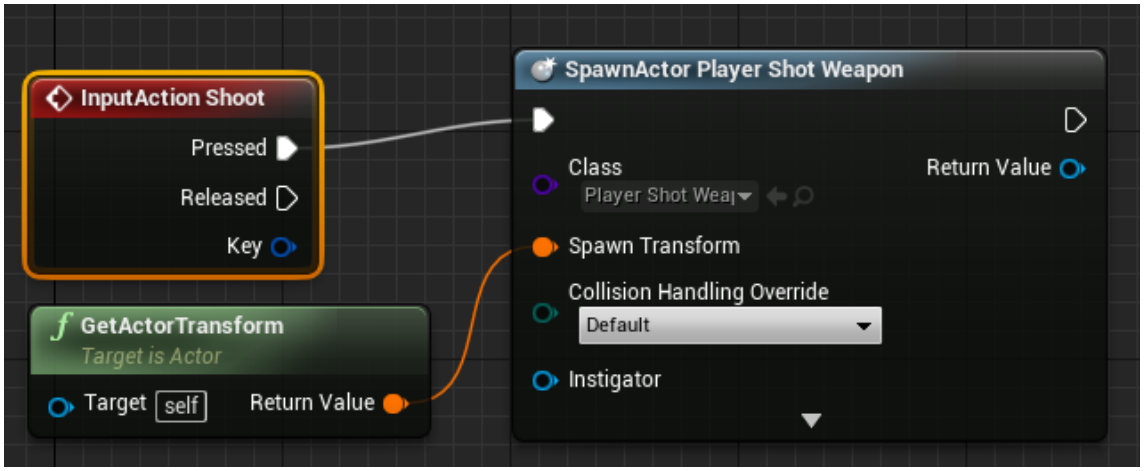
弾を発生させよう

UE4 はこういうイベントからノードを引っ張ることで、処理を追加していけます。ひとまず Pressed からノードを伸ばして、SpawnActorFromClass って書いて接続してください。

で、クラスの名前を弾の名前「Player Shot Weapon」にします。



実はこれだけだとどこから発射していいか分からずにエラーが発生しますので、今度は SpawnTransform からノード引っ張って、GetActorTransform を接続します。



これにより、プレイヤーから弾が発射される状態になります。

左クリックでプレイヤーから発射されるので確認してみてください。

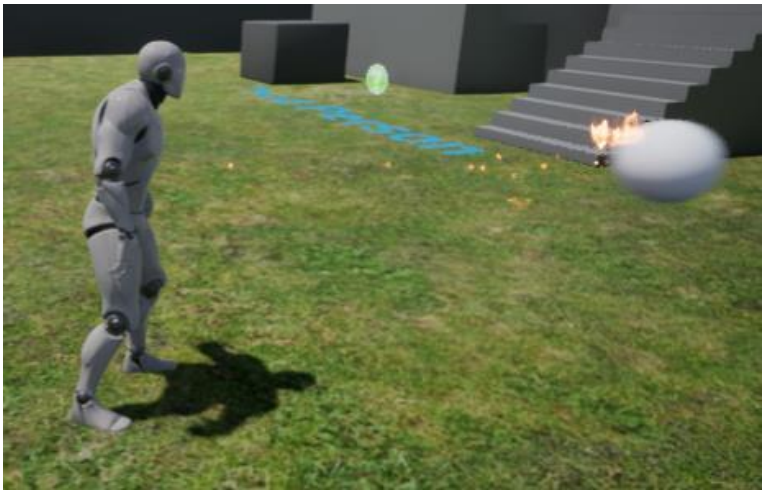
発射されない人はどこか間違っていますので、今までの流れをもう一度確認してみてください。

ファイアーしようぜ

ちょっと寂しいので球体の周りを燃やしてみましょう。もう一度 PlayerShotWeapon をダブルクリックしてください。
今度は ParticleSystem を追加します。

もちろん追加しただけじゃ、何も出ません。
どんなパーティクルにするのか選択しないといけません。今度は ParticleSystem を選択したままで右側の Particles の Template のなし▼をクリックして、P_Fire を選択します。

で、一度戻って再生してみると…



なんかファイアーの小さいのがついてますね。気に入らない人はファイアーの大きさを大きくしたり、複数のパーティクルを入れることでもっと派手にしていってください。ボールが邪魔だったら消してしまっても構いません(炎で弾の場所は分かりますので)

こんな感じになります。ピンポン玉じゃなくて鉄球みたいにしましょう。
Sphere って書かれてるところをクリックすると右側にマテリアルが出てきますので

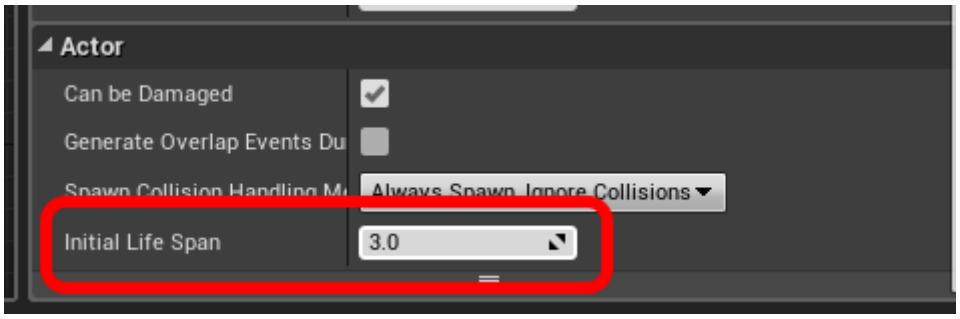
「M_Metal_Brushed_Nickel」を選んでください。

鉄球っぽくなると思います。

弾の寿命を設定しよう

ゲームにおいて弾丸がいつまでも残っていると色々と問題あるので弾の寿命を設定しましょう。
3秒くらいで消滅するようにしましょう。

ショットのルート(一番上)を選択した状態で詳細の下の方にInitialLifeSpanという項目がありますのでこれを0.0→3.0にしましょう。3秒くらいで消えるようになります。

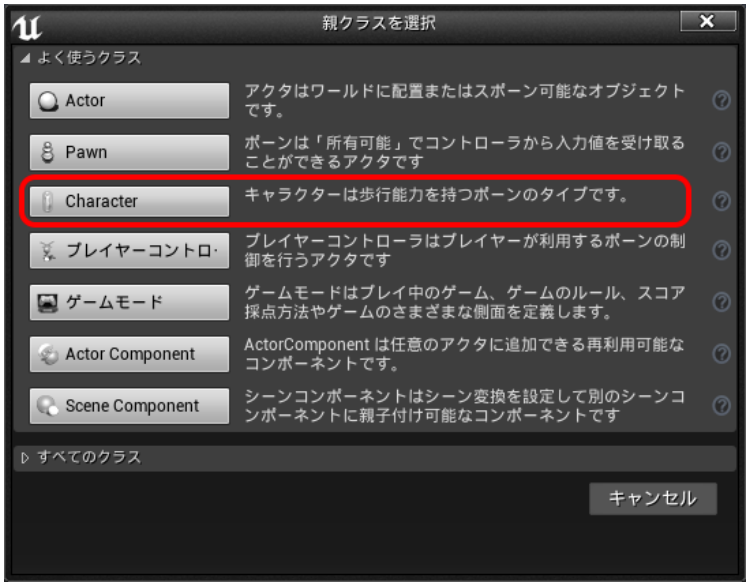


かなり下の方にあるので、ディスプレイによっては結構スクロールする必要があると思います。

さて、次は敵を作っていきましょう。

敵(マト)を作ろう

コンテンツブラウザ→新規追加→ブループリントをクリックしてください。



その中の Charater をクリックしましょう。

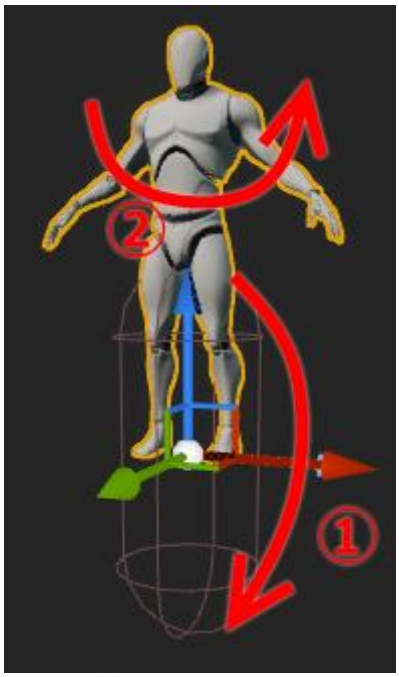
NewBluePrint ってのができますので、名前を enemyBP と名前を変更してください。

次はその enemyBP をダブルクリックしてください。

enemyBP の中に入って、左側に Mesh(継承)ってあると思うんで、それ選択してください

そしたら右側に Mesh 設定出てくるんで、SK_Mannequin を選択します。

そこでオブジェクトを見てみると、カプセルあたり判定と比較して浮き上がっていると思いますので、おろしてあげます。あと、メッシュの向きがオブジェクト方向と異なっているので、90°回転させてあげます。

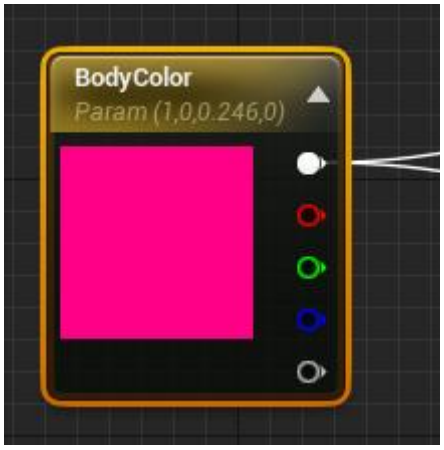


さて、このままだと自分と敵の区別がつかないので、見た目をちょっと変更します。いったん

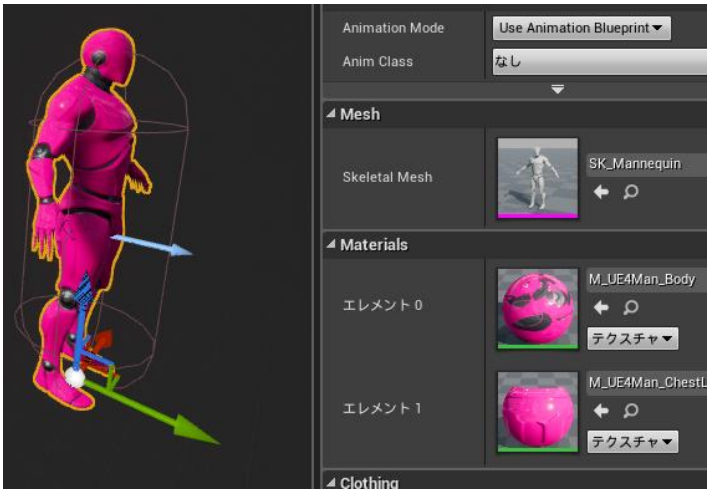
このマテリアルをコピーして改造しますので、一旦コンテンツブラウザに戻ってください。

で"M_UE4Man"で検索し、こいつらをコピー。色を敵っぽくします。

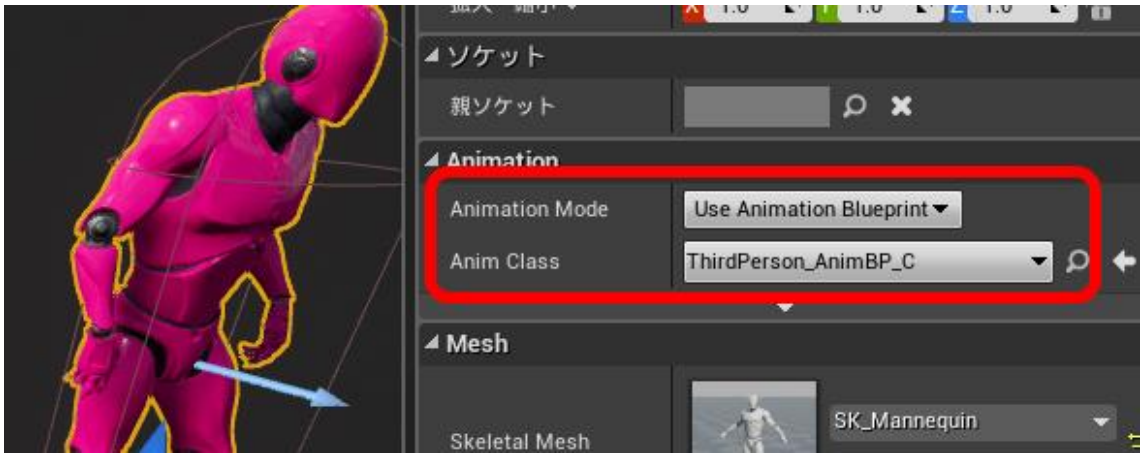
僕は赤っぽくしました。



色の変更の仕方は、左上くらいにある「BodyColor」をいじくります。好きな色に変えてください。



こんな風になってればOK。ただ、今のままだとデクノボーなのでアニメーション設定します。



とすることで、プレイヤーと同様のアニメーションをするようになります。
ということで、また元の画面に戻って、画面上にこのオブジェクトを配置してみましょう。

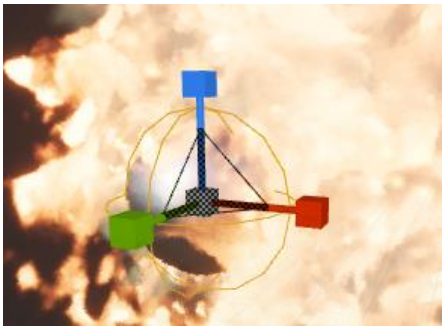
楽しくなってきました。

あたり判定

弾と敵のあたり判定を作っていきます。敵に当たったら敵が消えるようにしましょう。

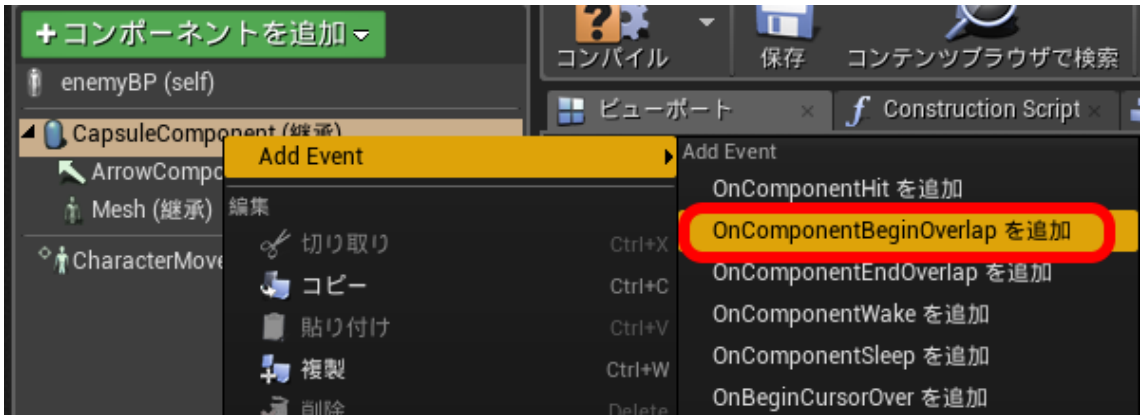
弾のあたり判定

PlayerShotWeapon を開いてコンポーネント追加→SphereCollision を追加しましょう。あたり判定がワイヤーフレームで表示されますので、大きさが気に入らない場合は拡大縮小で修正しましょう。



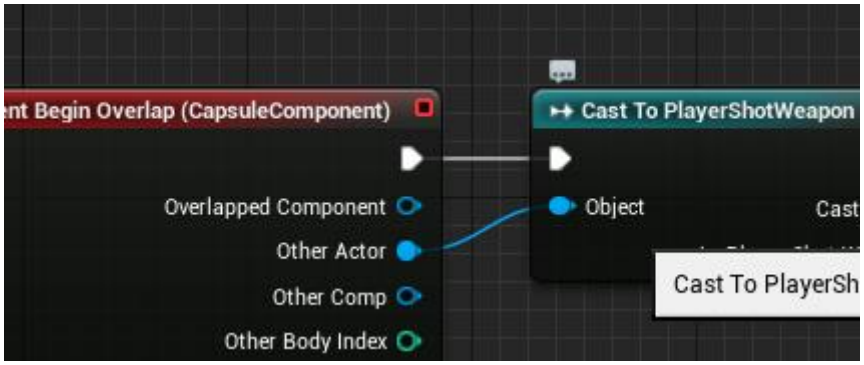
敵のあたり判定

それでは次にさっき作った敵のクラスを開いて、CapsuleComponent で右クリック→AddEvent →OnComponentBeginOverlap を押してください。

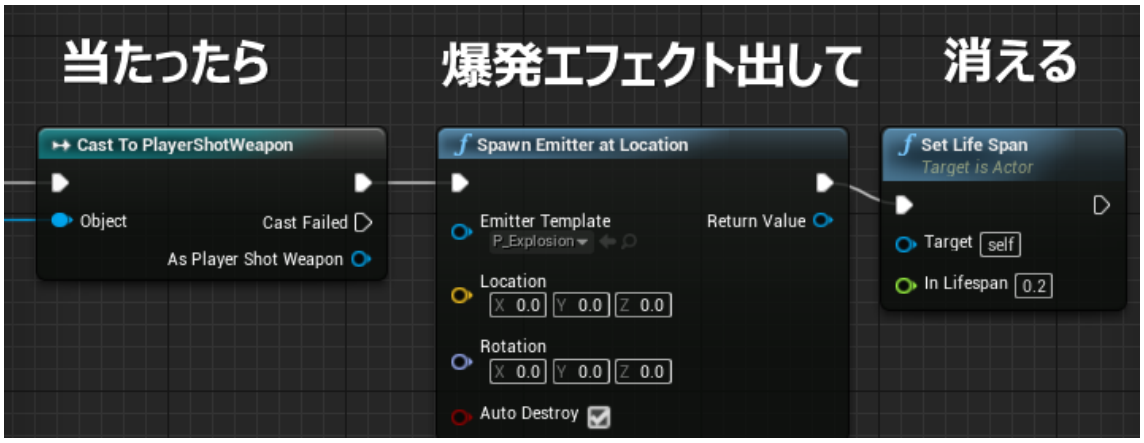


で、出てきたイベントから引っ張って「CastToPlayerShotWeapon」を選択

さらに OverlappedComponent と Object をつなぎます。

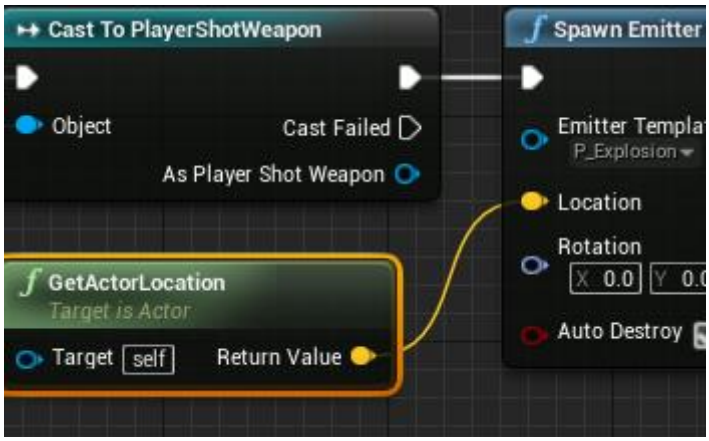


これで弾とのあたり判定が取れますので、あとはそれに対するアクションをとるだけです。



これで OK。できたら実行してみて、敵に弾を当ててみましょう。

あれ？爆発エフェクトが出てない気がするよ？
という事で、敵の座標をさっきの SpawnEmitter に放り込んであげましょう。
GetActorLocation でとってこれますので、SpawnEmitter の Location から引っ張って作りましょう。



そろそろ慣れてきたかなーと思うので、少し説明を簡易にしていきます。

弾の物理的あたり判定を無効にする

今はまだ弾の物理的なあたり判定が残っているので、自分や敵の座標がズレたりしますので、これを解消します。
PlayerShotWeapon を開いて、Sphere のコリジョンを NoCollision にします。また、SphereCollision の Collision を OverlapOnlyPawn にします。

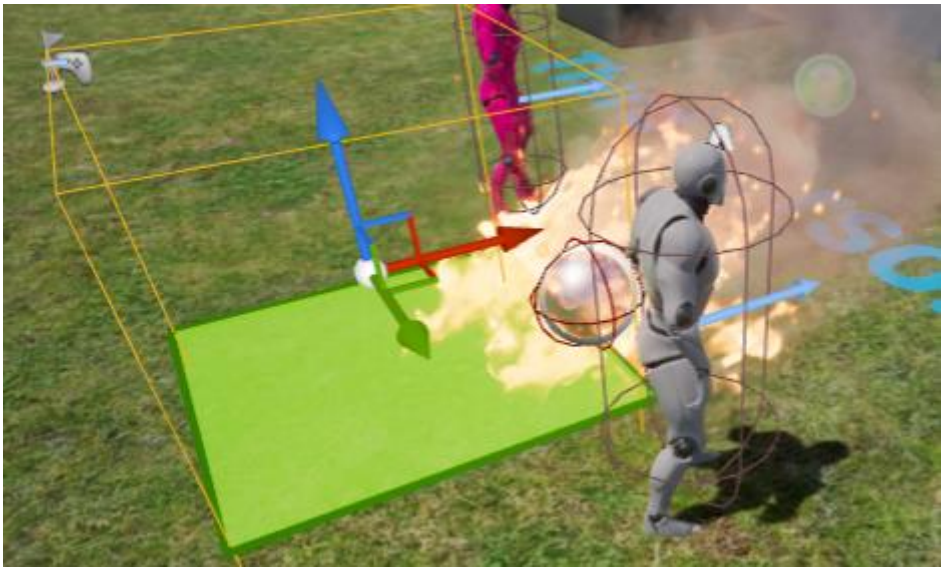
さて、ここまでやると、良い感じで敵に弾を当てた時の反応ができてきたと思います。

AI

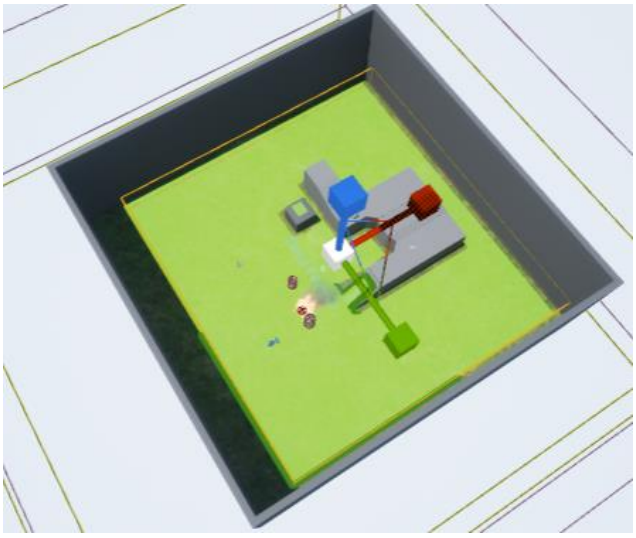
ナビメッシュの配置

このままでは敵もアホなままなので面白くありません。自律的に動くようにしてあげましょう。まず、敵の動きを制御する準備として、ナビゲーションメッシュをフィールド上に配置してみます。

配置したら P を押してみてください。索敵範囲が緑色で表示されます。もし緑色にならない場合は少し下に下げてください。



とはいえ索敵範囲がこれでは使い物になりませんので、拡大縮小でフィールド全体に広げます。



これで準備完了。
次はこれに沿って動くように AIController とビヘイビアツリーを作成します。

AIController とビヘイビアツリーの作成

AIController は新規作成→ブループリントクラス→全てのクラス→検索→AIController を選

択します。

名前を EnemyAIController にでもしておきましょう。

次にビヘイビアツリーです。

新規作成→AI→ビヘイビアツリー

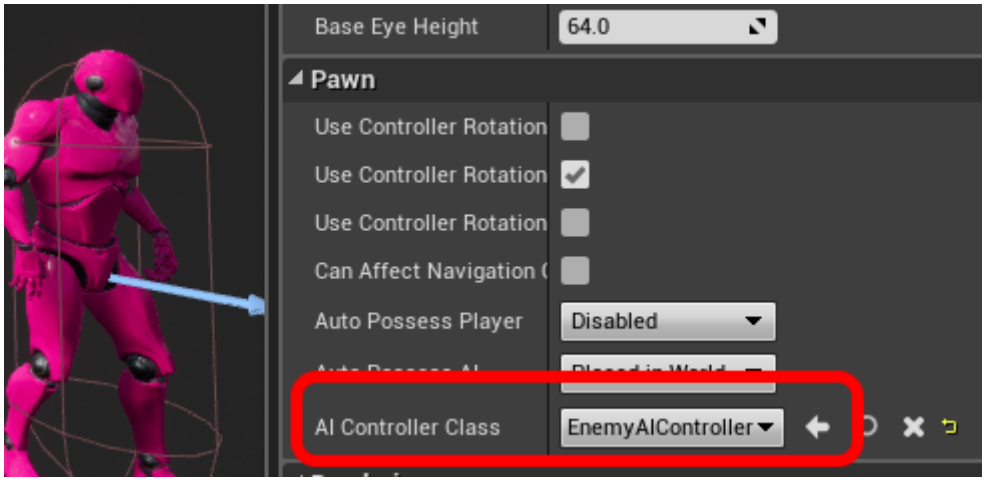
これも名前を EnemyBT とでもしておきます。

次に AI がビヘイビアツリーによって動くようにしていきます。AIController を開いて、Event BeginPlay をダブルクリックして、ブループリントを立ち上げます。

ここで BeginPlay から引っ張って、Behavior と入力して RunBehaviorTree を選択。BTAsset を EnemyBT にします。

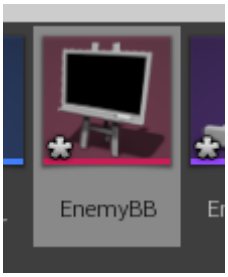
AIController と BehaviorTree の関連ができたので、Enemy 本体に関連付けます。Enemy クラスを開いて

Pawn→AIController を EnemyAIController にします。



とはいえ、残念ながらこのままではまだ動きません。
ビヘイビアツリーに何も設定してないからです。再び EnemyBT を開きます。
そこで新規ブラックボードを作成します。

そうすると勝手に BlackBoard ができてますので、名前は EnemyBB とでもしてやります。



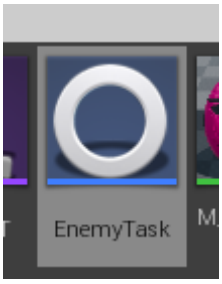
こいつは何のためにあるとかというと、索敵の際の変数を保管しておくところとでも思っておけばいいです。例えば一時的なプレイヤーの座標を保持しておくとかですね。

この EnemyBB をダブルクリックして、変数を追加します。プレイヤーの座標という事で、敵からすれば TargetPoint ですので、TargetPoint という変数を作ります。座標を表すので型(Key

Type)は Vector にしてください。

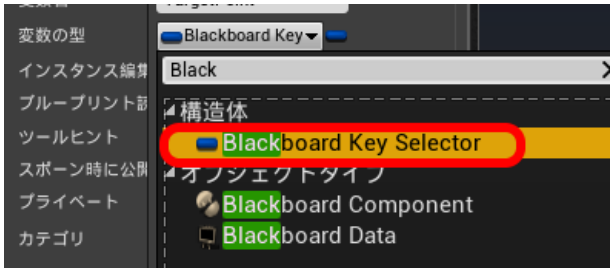
この段階では TargetPoint がプレイヤーの位置と関連付けられていないため、このまま動かしてもまだ敵は動きません。
プレイヤーの座標を TargetPoint に関連付けるために、EnemyBT を開いて「新規タスク」を作ってください。

こいつも勝手にできるので、名前を EnemyTask とでもしておきます。



今度は EnemyTask をダブルクリック。プレイヤーの座標を入れてあげます。変数として TargetPoint を追加

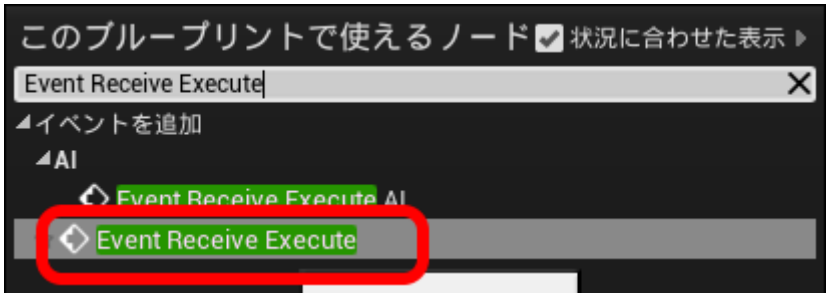
型がちょっとややこしいのですが、一覧に出てこないので、検索してください。BlackBoard と書けば出てきますが



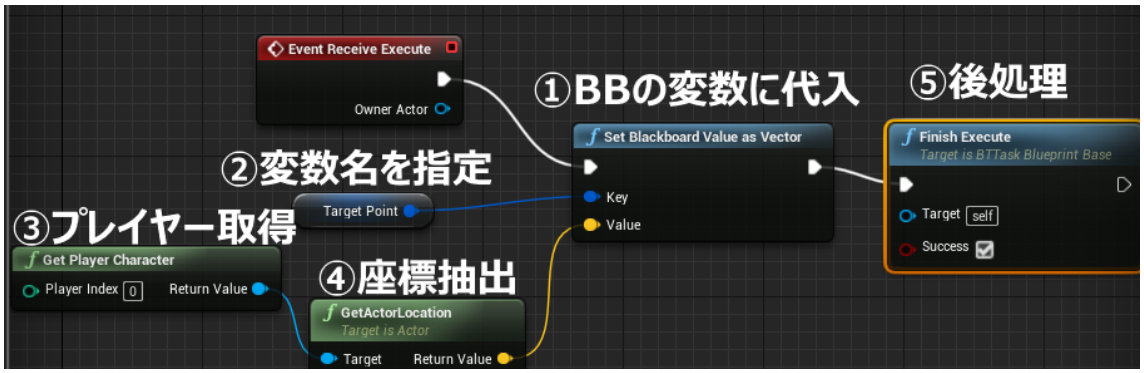
BlackboardKey Selecter を選択。

そしてその下のインスタンス編集可能にチェックを入れてください。

ここでブループリントを組んでいきます。右クリックして、Event Receive Execute で検索



これでイベントができますので、ここから組んでいきます。
TargetPoint にプレイヤーの座標を入れてあげるのが目的なので、やっていきます。



ちょっとややこしいので手順を書いておきます。

① 関連付けられている Blackboard の変数に代入するために Set Blackboard します。
② Blackboard の変数名を取得するために Key から引っ張って TargetPoint を指定
③ 何もないところで右クリック&GetPlayerCharacter します
④ プレイヤーオブジェクトからプレイヤーの座標を抽出して Blackboard にセット
⑤ 最後の後処理として、ここまで実行後に Success をオンにします。

さて、これでプレイヤーの座標が TargetPoint に入りました。

ビヘイビアツリーとブラックボードを関連付け、挙動を定義

これでやっとブラックボードとビヘイビアツリーを関連付けられます。EnemyBT を開いて、Blackboard Asset に EnemyBB を設定。
そうしたら TargetPoint(一時的なプレイヤーの座標)を使用することができるため、あとは敵の挙動を定義していくだけです。

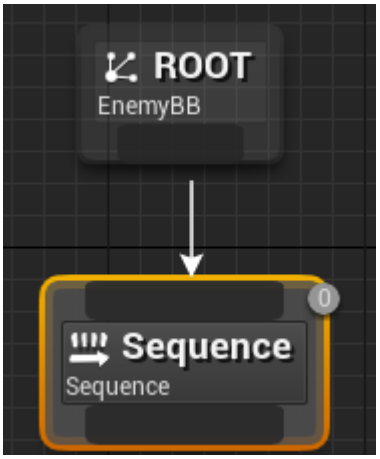
敵の動きとしては

① 索敵(プレイヤーを探す…座標を取得する)

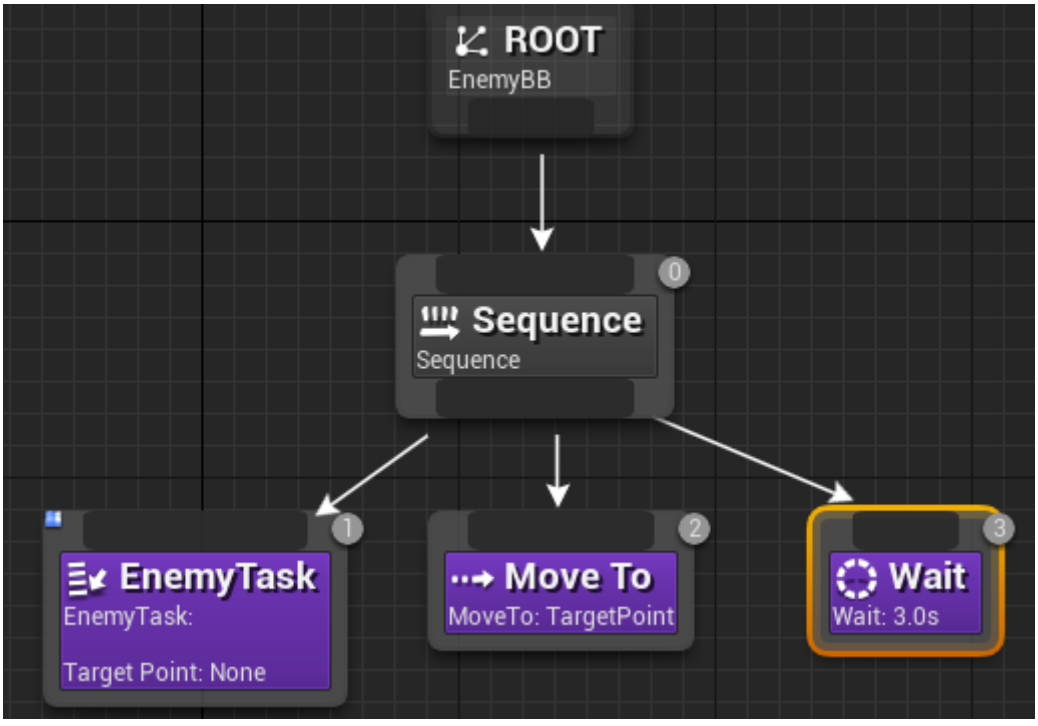
② 探した方向に(取得した座標に向かって)動く
③ 待ち

この「待ち」がないと割と無理ゲーになるので、待ちを入れています。この待ち時間しだいでゲームの難易度を調整できるでしょう。

こういう一連の動きを定義するには Sequence を使用します。Root から引っ張って Sequence を作ります。



Sequence は分岐とかせずに、一連の動作をまとめて行うためのものです。そして、先ほどの流れを Sequence から線を引っ張って定義していきましょう。



Wait のデフォルトが 5 なのですが、簡単すぎるかなと思って 3 にしています。これで実行する

と、プレイヤーを追っかけてくるはずです。

「あれ？でもこれって、ナビメッシュ関係してるの？プレイヤー追っかけてくるだけじゃん」と思った人は、障害物の後ろに逃げてください。

難なく障害物を避けてこちらに来ると思います。ナビメッシュがないと障害物に引っかかっちゃうはずです。

ゲームらしくしていこう

ここまでで TPS の基本はできました。あとはもう少しゲームらしくしていきましょう。

ライフを作ろう

ライフを作ります。プレイヤーの変数として、ライフを追加します。**heath という変数名で Flaot にします。**

※「インスタンス編集可能」にチェックを入れておくのを忘れずに。

体力を可視化するために、体力ゲージを作ります。こういう 2DUI 的なものはウィジェットを使用して実装します。

新規作成→ユーザーインターフェース→ウィジェットブループリント

名前を GameUI とでもしておきましょう。

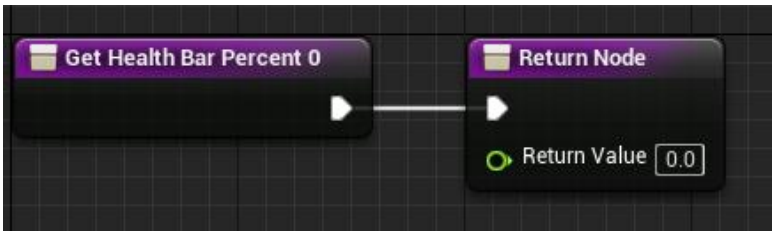
早速ダブルクリックして編集していきます。

パレット→一般→ProgressBar をドラッグアンドドロップしてください。

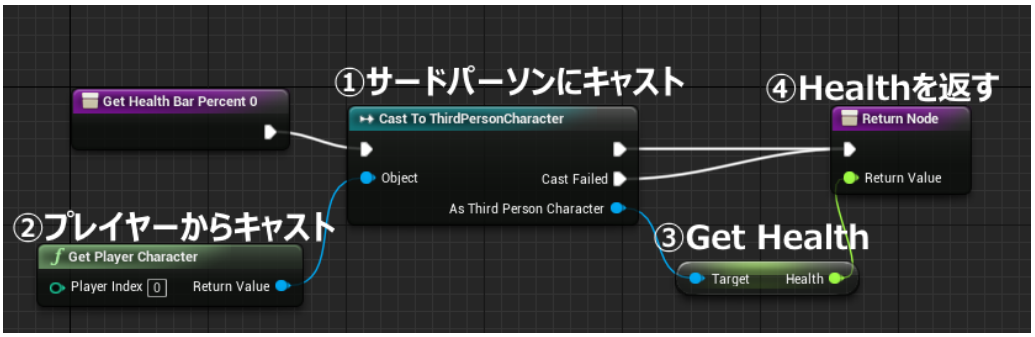
名前は HealthBar とでもしておきましょう。大きさと色(テクスチャも OK よ)は好きに調整してください。

右側に Progress って出てきていると思いますんで、バインドってのをクリック

バインディングを作成します。
ここでまたもやブループリントが出てきます。

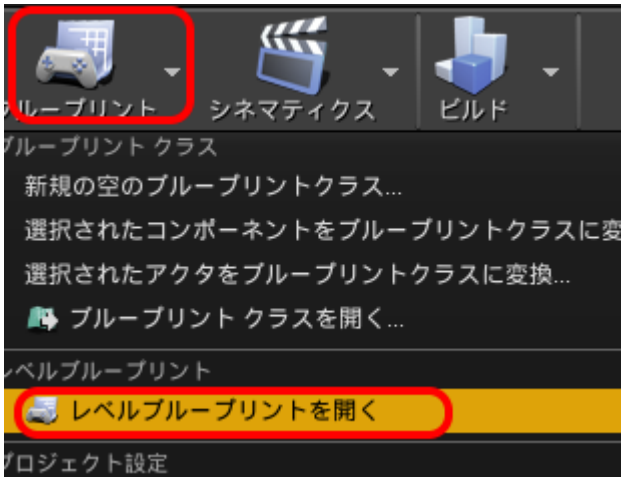


こんなの出てくるので、この間に色々と処理を書いていきます。



②および③は、青いところから引っ張らないと出てこないので注意。また、プレイヤーに health を作り忘れてると③が出ないので注意。

このままではUIが画面に反映されないのでレベルブループリントを開きます。



最初は何も書かれてないので
このようにします。

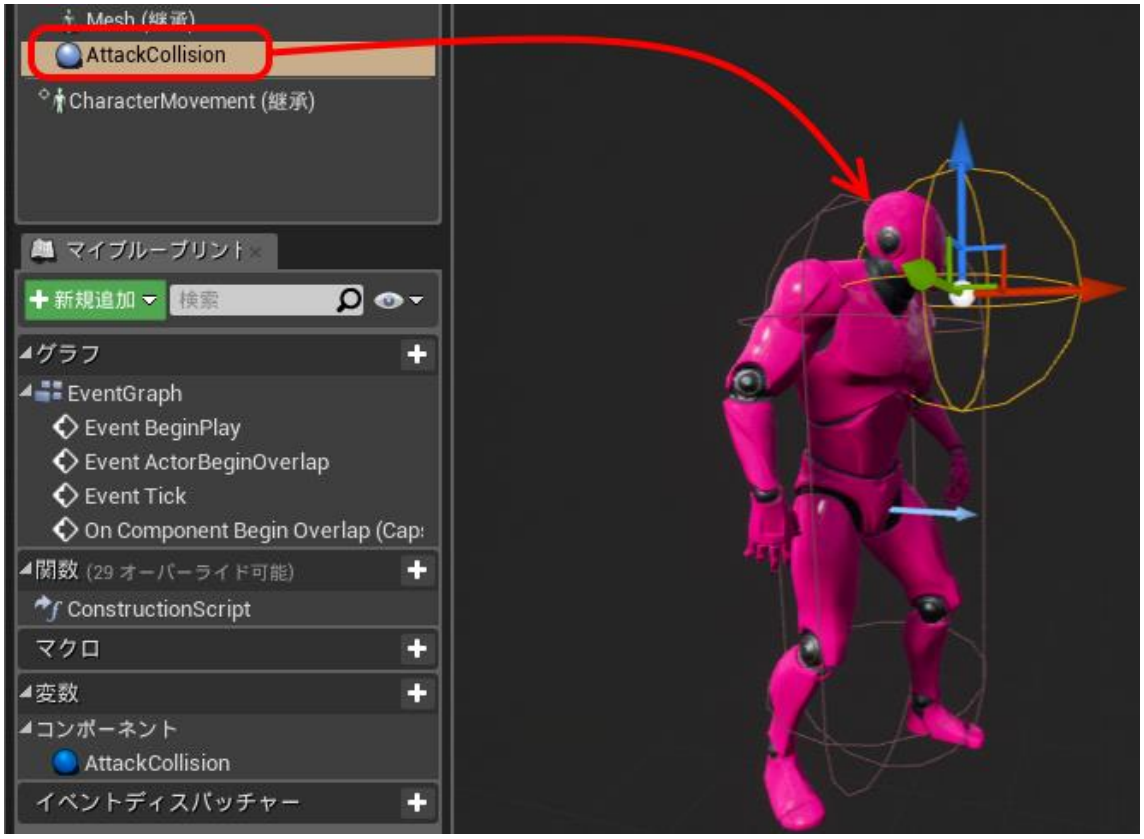
ここまでやれば画面上に設定した体力ゲージが表示されます。

先ほども書きましたが、テクスチャも OK なので

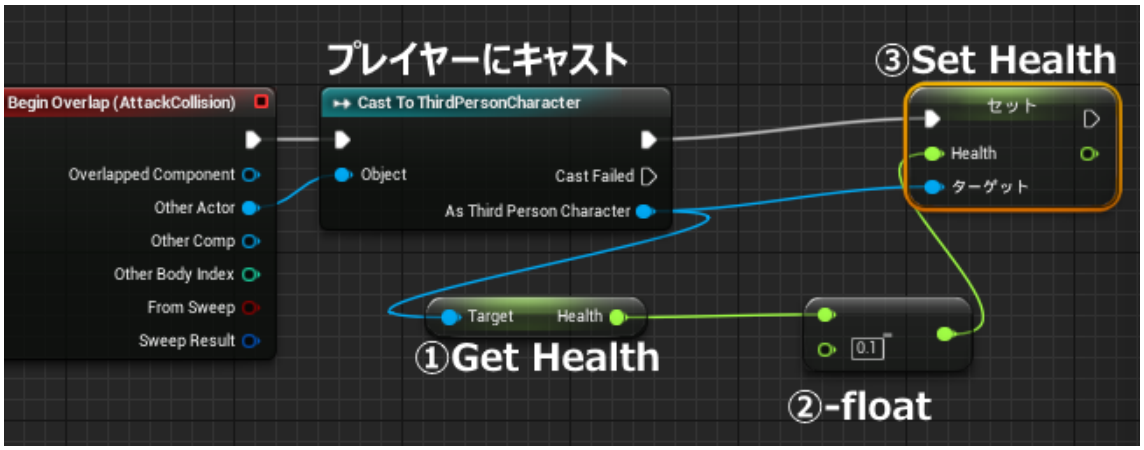
こういうのも可能です。そこはお好きにしてください。

ライフバーである以上は減らさなければならないので、ダメージ設定していきましょう。

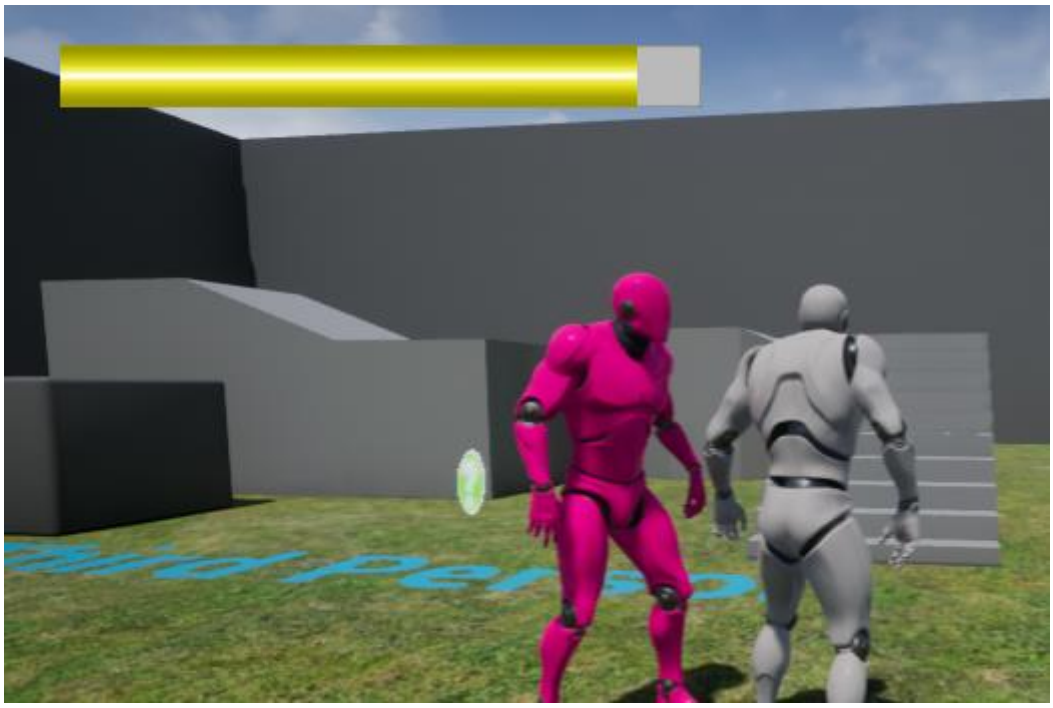
敵に攻撃判定を付ける



敵に SphereCollision をつけて、鼻先にくっつけてあげます。これがプレイヤーと衝突したら、プレイヤーにダメージが入るようにしましょう。
右クリックして、ConComponentBeginOverlap イベントを追加します。このあたり判定が当たった相手がプレイヤーだったら、ライフを削る仕組みを作りましょう。



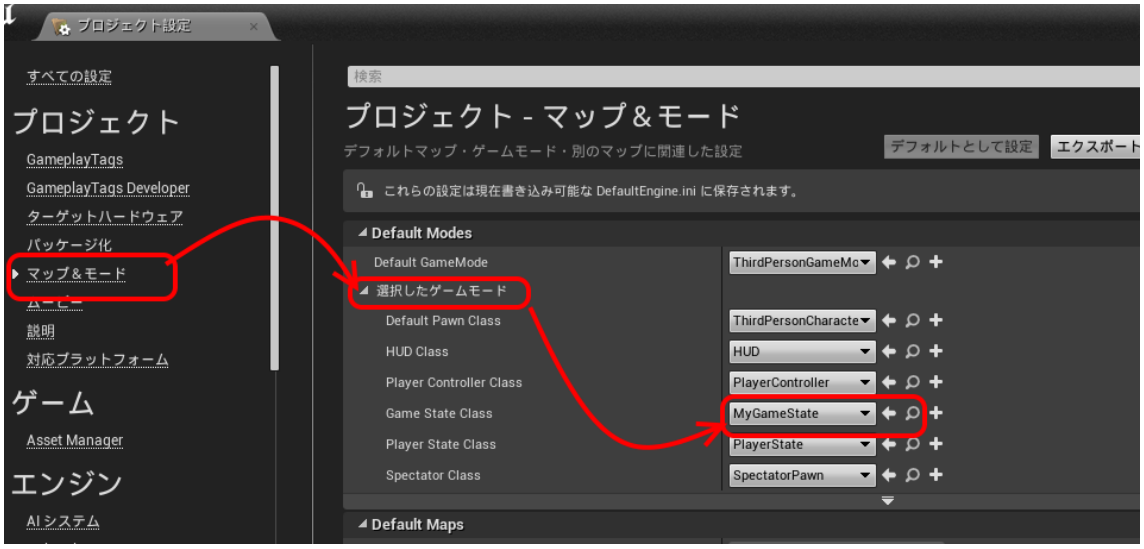
もし、どこかで間違ってなければ相手がぶつかるたびに、体力ゲージが削られていくはずです。



スコアを表示しよう

まず、スコア管理用のブループリントを作る必要があります。
新規作成→ブループリント→検索→GameState

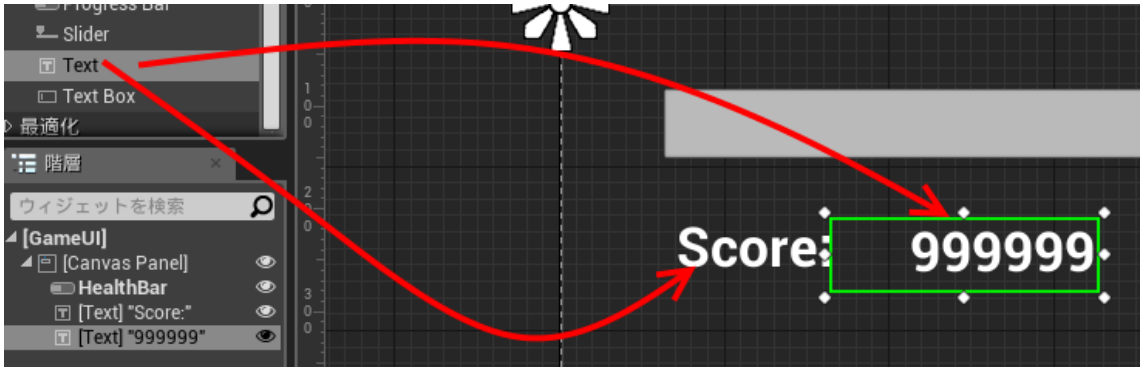
名前を MyGameState 等にしておきましょう。
で、これをプロジェクトそのものに関連付けるために編集→プロジェクト設定→マップ&モード
DefaultModes の「デフォルトゲームモード」を展開し、GameStateClass を MyGameState にしましょう。



再び MyGameState に戻り、スコア用変数 Score を追加します。型は Integer 型にします。

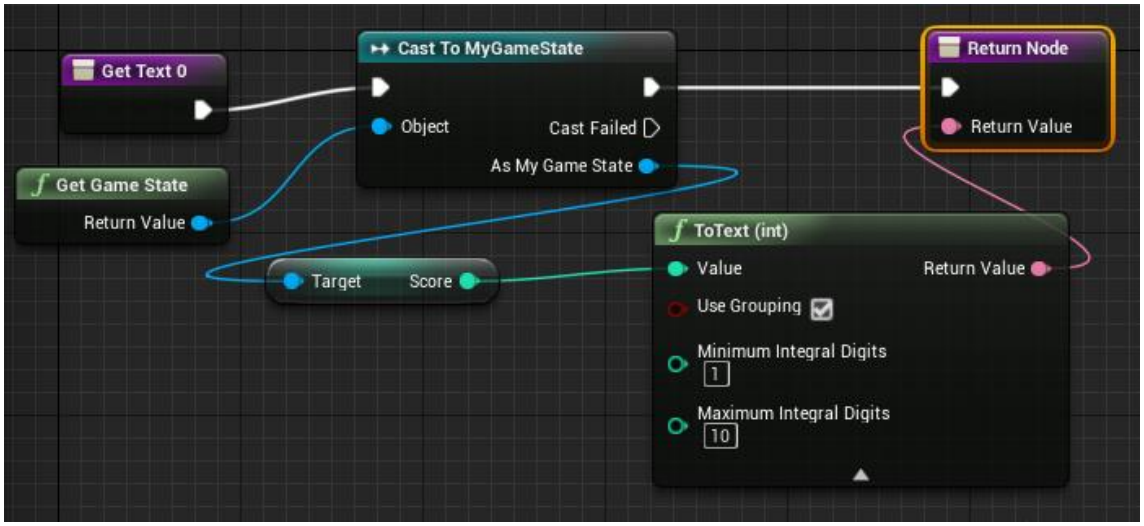
毎度のことですが、インスタンス編集可能のチェックをしておきます。

今度は GameUI を開いて、スコア表示の準備をします。Text をそれぞれ配置し、Score:と 99999 と設定します。



次にこの 99999 のほうは、Score とのバインディングを行うため、バインディングの追加をします。

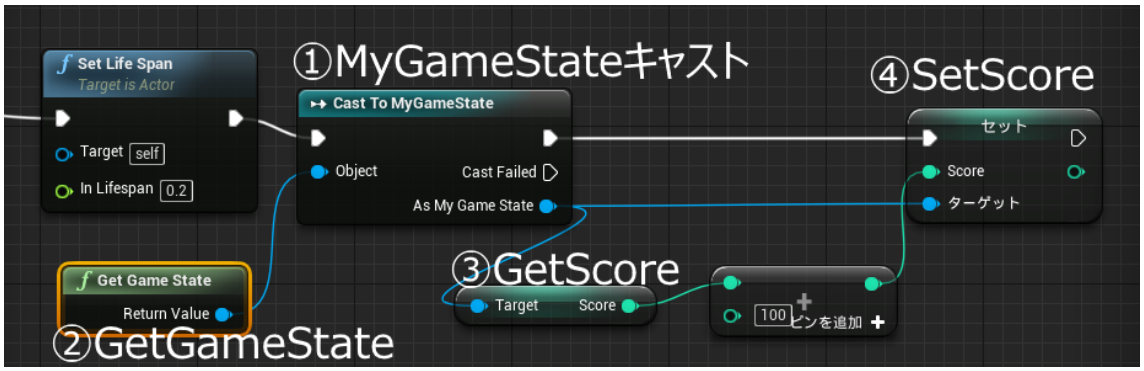
ちょっとごちゃっとして申し訳ないが、このようにしてください！



なお、ToText の部分で Min と Max があって、これ最初勘違いしたんですが、よく見ると数値の最大値と最小値じゃなくて、**桁数の最大値最小値で**すね。ですから 1～10 くらいで十分です。

スコアゲット

次に実際に敵を倒したらスコア貰えるようにしましょう。
EnemyBP のブループリントの最後の部分に以下の処理を追加します。



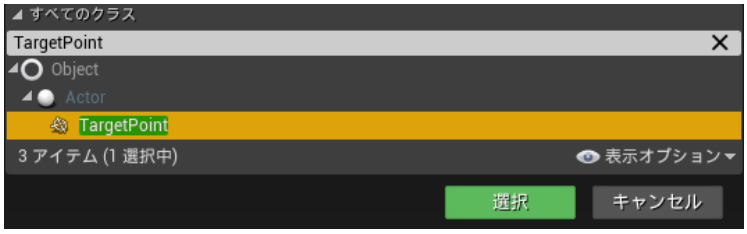
これで敵を倒したら、スコアが入るはずです。やってみましょう。

上の絵のように、スコアが変われば成功です。

※余談ですが、暑さのせいか、ここまで作る途中に 2 回くらい落ちました。こまめにセーブしましょう。

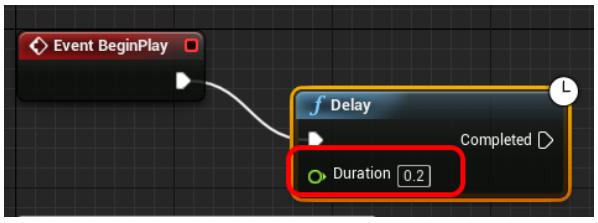
敵を増やそう

このままでは敵が一体だけでちょっと寂しいので無限湧きさせてみましょう。
新規作成→ブループリント→検索→TaretPoint で作ります。

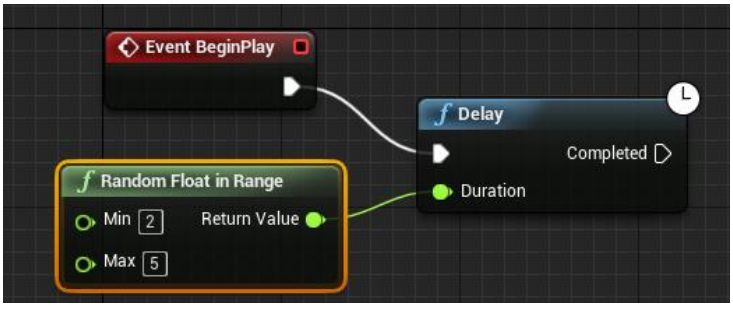


名前は EnemySpawnPoint にしましょう。
こいつをダブルクリックしてブループリント開いて、EventBeginPlay をダブルクリック。ここ

から始めていきます。敵をぽんぽん出現させていきたいのですが、制限を設けないと大変なことになるので一定時間ごとに出現させてみます。BeginPlay から引っ張って Delay を作ります。これは一定時間待って次のノードを実行させるものです。

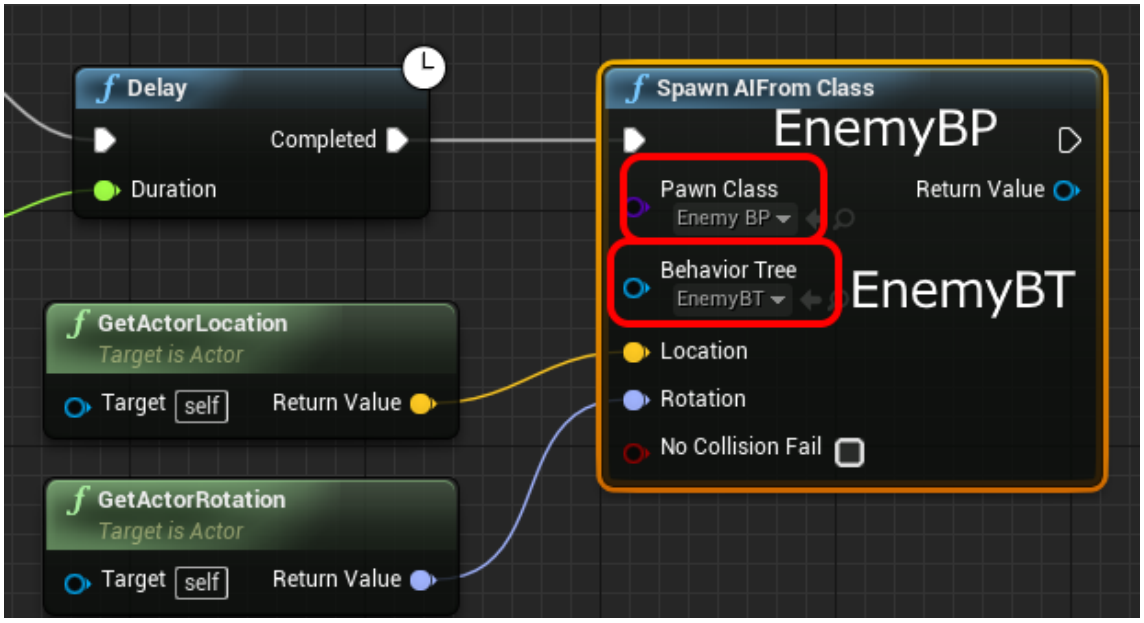


デフォルトだと 0.2 秒でちょっと早すぎますね。3 秒くらいにしておきましょうか。
でも、例えばランダムに出現させたければ、Duration から引っ張って、RandomFloatInRange



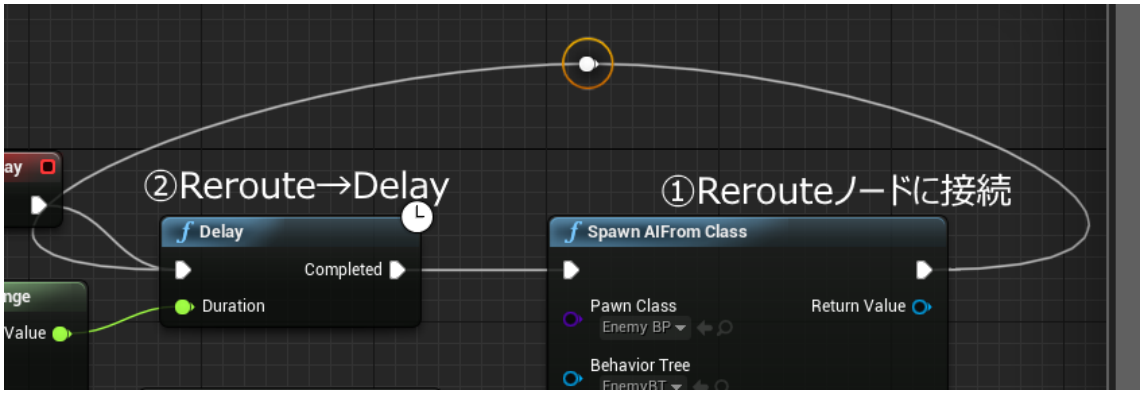
などを使用すれば範囲内のランダムで実行することができます。

そして、もちろん実行するのは、敵の出現。というわけで、Delay の Compelete から引っ張って



このようなブループリントを構築します。
AI つきのキャラクターを出現させたいので Spawn AIFrom Class を使用します。で、Pawn Class に EnemyBP、そして Behavior Tree に EnemyBT を設定します。
そして出現場所は、配置する TargetPoint と同じ座標にしたいので、GetActorLocation を設定。GetActorRotation は別に設定しなくてもいいです。

ただ、今のままだと TargetPosition ごとに一体しか出てこないので、無限湧きにするためにループを付け加えます。SpawnAIFromClass から引っ張って Reroute ノードを作り、そしてまたそれを Delay に接続してループ状態にします。



これにより無限湧きを実現できます。

こうやって作った TargetPoint をフィールドの 4 隅に配置しましょう。
アホみたいに敵が湧いてきます。

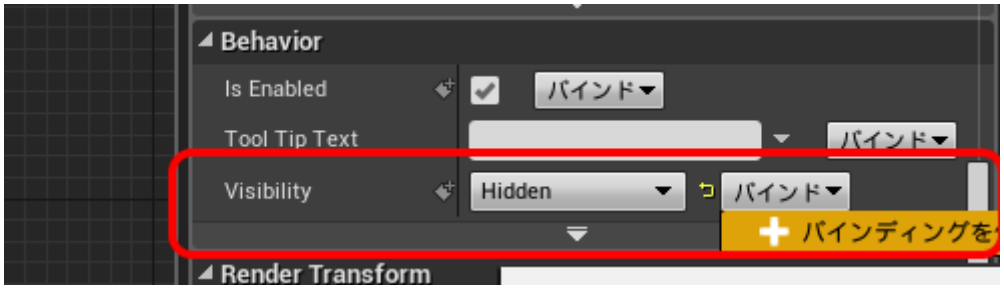
ゲームオーバー

現状だと体力がなくなっても生き続けるので、ゲームオーバーを実装します。

GameUI を開いて、真ん中にテキストを配置…GameOver!!!!というテキストにします。デカくしたいときはフォントサイズを大きくしてください。色も絶望感の赤にしましょう。

でもこのままだと、ゲーム開始時点からゲームオーバーなので、これの可視フラグもバインドで制御しましょう。

右側に Visibility とあるので、デフォルトを Hidden にして、バインディングを作成しましょう。

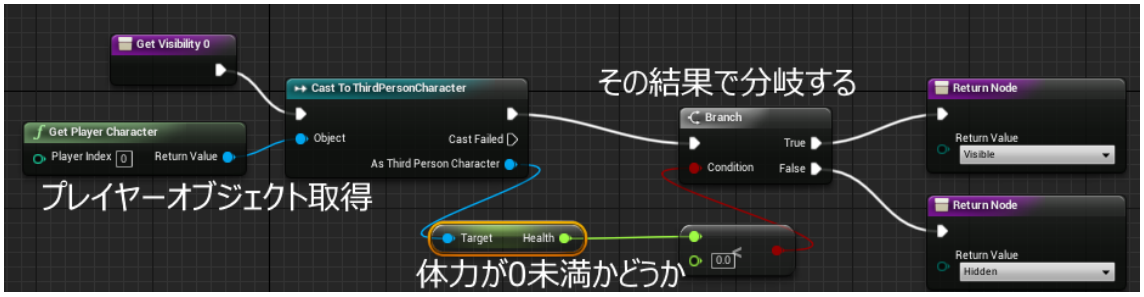


バインディング作成画面に入ったら、変数を追加します。Visible という名前で、

型は ESlateVisibility です。

そしてこの変数をプレイヤーの体力を元に変化させるようにします。

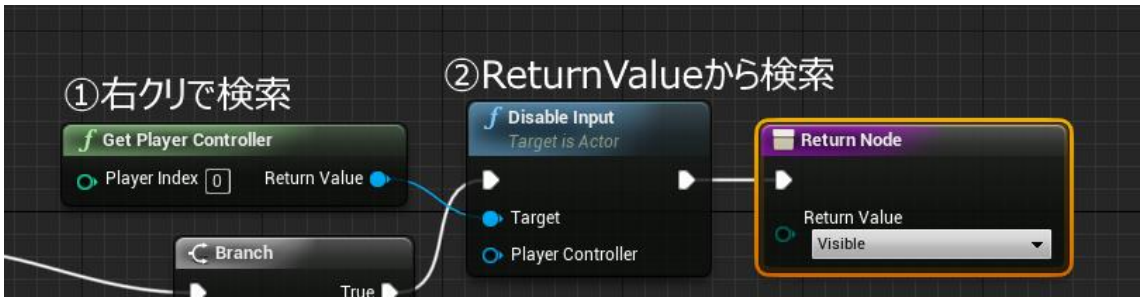
プレイヤーの体力が 0 以上だったら表示しない(Hidden)。0 未満だったら表示する(Visible)と。



なお、Branch は Condition として入ってきた評価式が True か False かで実行先を分岐するものです。

ここまで設定が完了すれば体力があるうちは表示されず、体力がなくなったら GameOver になります。

現状だと、死んだ後でも動き続けますので、先ほどの部分に動かせなくする処理を入れます。

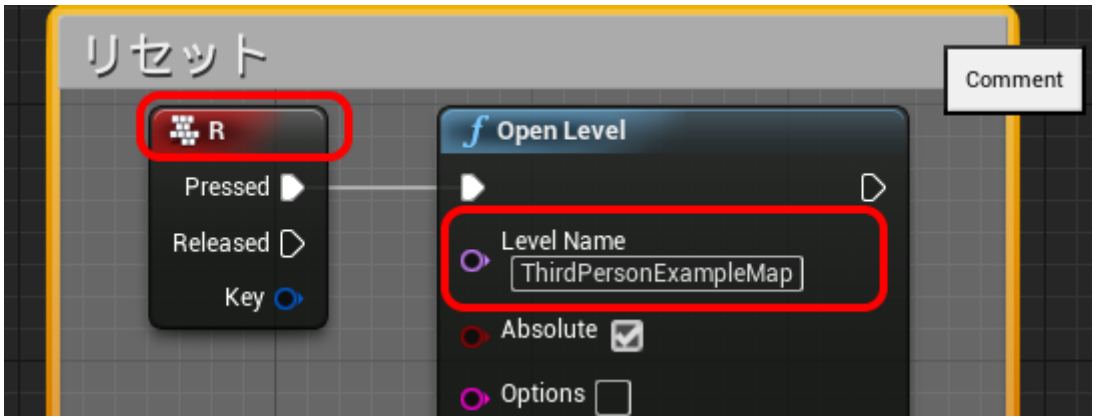


何もない部分で右クリで検索「Get Player Controller」そしてその Return Value から検索し、「Disable Input」これを Branch の True から結び付け、反対側を ReturnNode の Visible 側にくっつけます。これでゲームオーバー後は操作不能になります。

リセット機能

このままだと、ゲームオーバーで何もできなくなるため、リセット機能を追加します。R ボタンで最初の状況に戻るものです。

レベルブループリントを開き、右クリでインプット→KeyboardEvents→R を選択。OpenLevel を呼び出し、自分自身のレベルを呼び出します。この時の名前はプロジェクト設定で GameDefaultMap に設定されている名前を使用してください。



これでようやく終了です。
お疲れ様でした。