

算数ゲーム

UE4のブループリントでプログラム文法を1日で体験する

v1.4

Index

1. 作成するゲーム内容、得られる知識
2. プロジェクトを作成する
3. ブループリントでHello Worldを画面に表示する
4. ブループリント(BP)で足し算ゲームを作成する
5. 計算方法(足し算、引き算、掛け算、割り算)をランダムに修正する
6. UMGの作成・表示（ウィジェットブループリント）
7. UMGからゲームを実行できるようにする
8. テスト項目票の作成、デバッグ
9. パッケージ化

- 1. 作成するゲーム内容、得られる知識
- 2. プロジェクトを作成する
- 3. ブループリントでHello Worldを画面に表示する
- 4. ブループリントで足し算ゲームを作成する
 - 4.1 新規レベル (RandomPlusGame) を作成・保存
 - 4.2 ブループリント(BP_RandomPlusGame)を作成
 - 4.3 足し算の実装
 - 4.4 足し算の引数を変数化
 - 4.5 足し算を関数化
 - 4.6 ランダムの数値を足し算する関数(RandomCalculation)を作成
 - 4.7 キーボード入力で答えを入力
 - 4.8 正解判定
 - 4.9 算数ゲーム(足し算のみ)化
- 5. 計算方法(足し算、引き算、掛け算、割り算)をランダムに修正する
 - 5.1 新規レベル (RandomCalcGame) を作成・保存
 - 5.2 BP_RandomPlusGameを複製して、BP_RandomCalcGameを作成
 - 5.3 引き算、掛け算、割り算を実装
 - 5.4 関数RandomCalculationの修正
 - 5.5 算数ゲームの計算方法のランダム対応
- 6. UMGの作成・表示 (ウィジェットブループリント)
 - 6.1 新規レベル (DispWidget) を作成・保存
 - 6.2 画像ファイルのインポート
- 6.3 ウィジェットブループリント(WBP_MathGameOutline)を作成
- 6.4 ウィジェットブループリントエディタについて
- 6.5 ウィジェットの配置
 - 6.5.1 Imageを使用してアウトライン背景を表示
 - 6.5.2 Textを配置して計算式を作成
 - 6.5.3 HorizonBoxを使用して、1~9のButtonを横並びに配置する
 - 6.5.4 Imageを使用して結果判定の画像を配置
- 6.6 レベルブループリントの編集
 - 6.6.1 ウィジェットを表示
 - 6.6.2 マウスカーソルを常に表示
 - 6.6.3 ESCキーでゲームを終了
- 7. UMGからゲームを実行できるようにする
 - 7.1 新規レベル (MathGame) を作成・保存
 - 7.2 WBP_MathGameOutlineを複製して、WBP_MathGameを作成
 - 7.3 背景をアウトラインがない画像に変更
 - 7.4 WBP_MathGameOutlineからWBP_MathGameを表示するようにレベルブループリントを修正
 - 7.5 ウィジェットブループリントの基本機能を実装
 - 7.5.1 ButtonのClickイベント
 - 7.5.2 TextのTextプロパティの値を変更
 - 7.5.3 Imageの画像切り替え
 - 7.5.4 Imageの表示/非表示

7.6 BP_RandomCalcGameの処理を参考にWBP_MathGameを実装

 7.6.1 BP_RandomCalcGameから変更するポイント

 7.6.2 BP_RandomCalcGameから変数、RandomCalculationを移植

 7.6.3 ボタンのクリックイベントをTextBlock_NumAのTextに設定

 7.6.4 計算とボタンの入力値を判定して結果画像と答えを表示

 7.6.5 計算と入力判定のループ化

8. テスト項目票の作成、デバッグ

 8.1 そのゲームちゃんと動くの？

 8.2 ゲーム内容からテスト項目の抽出

 8.3 テスト項目票を作成して、テストを実施

 8.4 ブループリント、関数単位のテストするためのデバッグ方法について

 8.5 ブラックボックステスト、ホワイトボックステスト

 8.6 作りたいゲームの作り方、工程に対応するテスト

9. パッケージ化

 9.1 配布用パッケージにするためのプロジェクト設定

 9.2 Windows(64ビット)でパッケージ化

 9.3 パッケージサイズをより小さくする

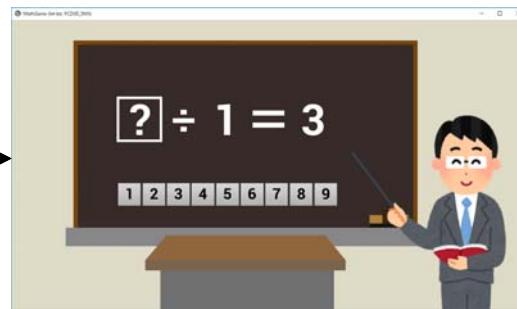
変更履歴

バージョン	日付	変更内容
V1.3	2017/10/24	変更履歴追加(V1.3~),UI用の画像インポート後のUI用画像設定について追加
	2017/11/01	UnrealEngine 4.18.0で動作確認 UE4 Style Guideについての記述を追記 列挙型 E_CalcType > ECalcType, 構造体 ST_CalcResult > FCalcResult 画像のファイル名 IMG_ > T_ 画像の変更箇所だけ記入 次のリリース時にキャプチャ撮り直し
V1.4	2018/7/3	Udemy公開用に説明文を追記

1.作成するゲーム内容、得られる知識

作成するゲーム内容

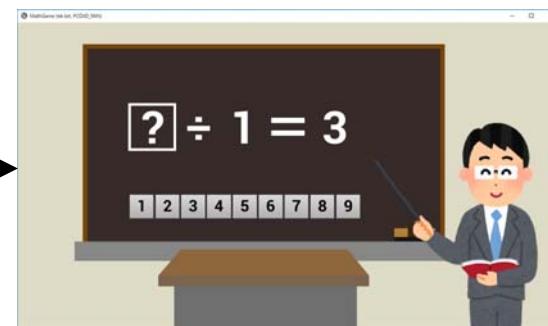
問題出力



数値A、数値Bの数字、
計算方法(+,-,x,÷)がランダム

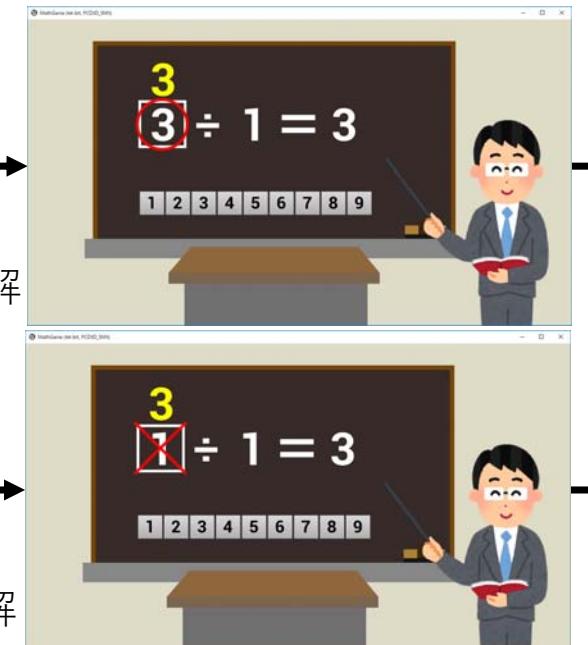
計算式の数値Aは?で
表示されていて分からぬ

入力



数値Aの数字と一致する
ボタン(1~9)をクリック

解答出力



正解

不正解

数値Aの値と解答に表示されるランダムで出した
数値が一致するかどうかで正解か不正解か分かる

ESCキー：ゲーム終了

繰り返し

得られる知識

- ブループリントの作り方（基本操作）
- 変数
- 計算ノード
- 関数
- If文(ブランチ)
- キーボード入力
- UIの作り方(ウィジェットブループリントのデザイン)
- ウィジェットブループリントの処理（ボタン入力、表示の切り替え）
- 設計書からのテスト項目作成
- デバッグ
- パッケージ化(Exeの書き出し)

使用するアプリケーション

- UnrealEngine4(Winsows 10) Version 4.17.2
- UnrealEngine4(Winsows 10) Version 4.18.0 動作確認済
- UnrealEngine4(Winsows 10) Version 4.19.2 動作確認済

UE4 Style Guideに準拠

Allar / ue4-style-guide

Watch 66 Star 351 Fork 105

Code Issues 1 Pull requests 2 Projects 0 Wiki Insights

An attempt to make Unreal Engine 4 projects more consistent <http://ue4.style>

blueprint naming-conventions unreal-engine-4 unreal-engine unreal unreal-marketplace unrealengine ue4 udk udk4 styleguide
style-guide linter linters lint linting

87 commits 2 branches 0 releases 4 contributors MIT

Branch: master New pull request Create new file Upload files Find file Clone or download

Allar Merge pull request #18 from akenatsu/typo/Section4.3 ...
Latest commit 474fcc a day ago

File	Commit Message	Time Ago
images	Beginning graph style	9 months ago
LICENSE	Update LICENSE	a year ago
README.md	fix from 4.2 to 4.3	2 days ago
marketplace-compatibility.md	Update marketplace-compatibility.md	3 months ago

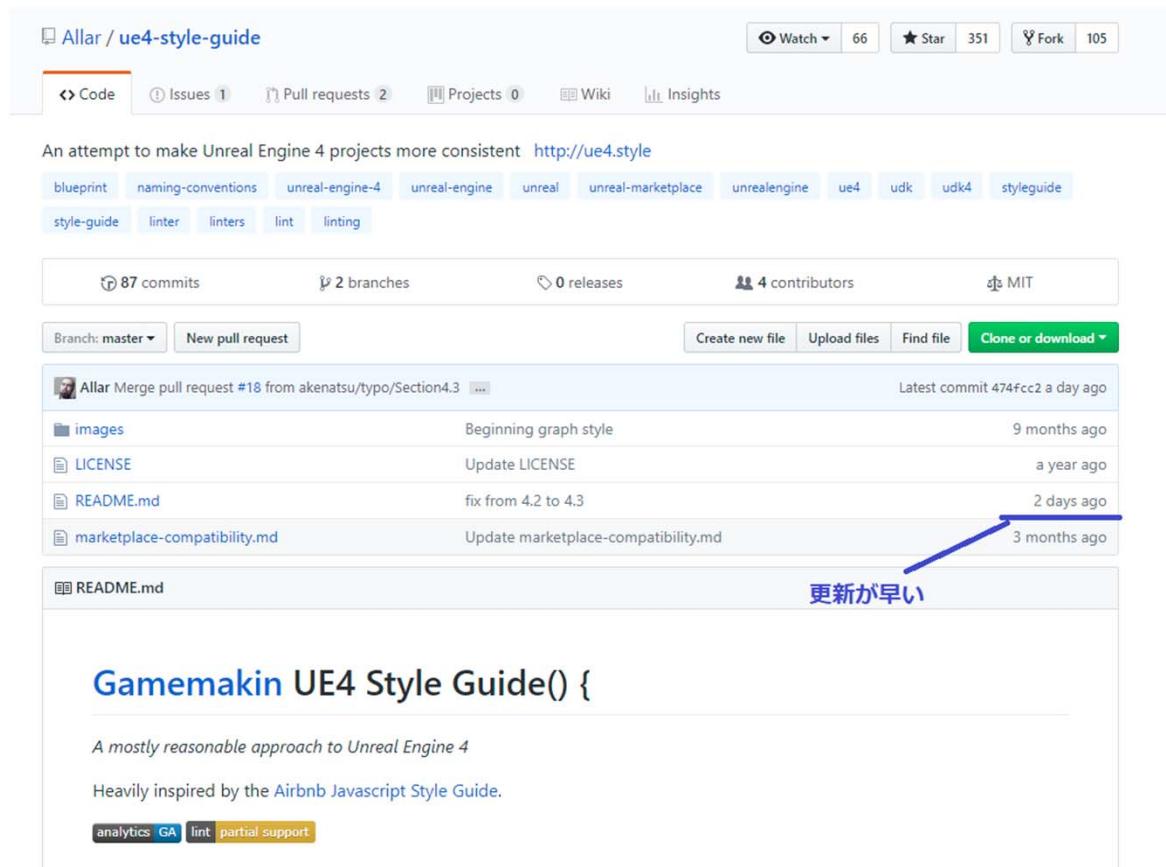
README.md 更新が早い

Gamemakin UE4 Style Guide() {

A mostly reasonable approach to Unreal Engine 4

Heavily inspired by the Airbnb Javascript Style Guide.

analytics GA lint partial support



ファイル名やフォルダ構成については
UE4 Style Guideに準拠するようにしています

ue4-style-guide

<https://github.com/Allar/ue4-style-guide>

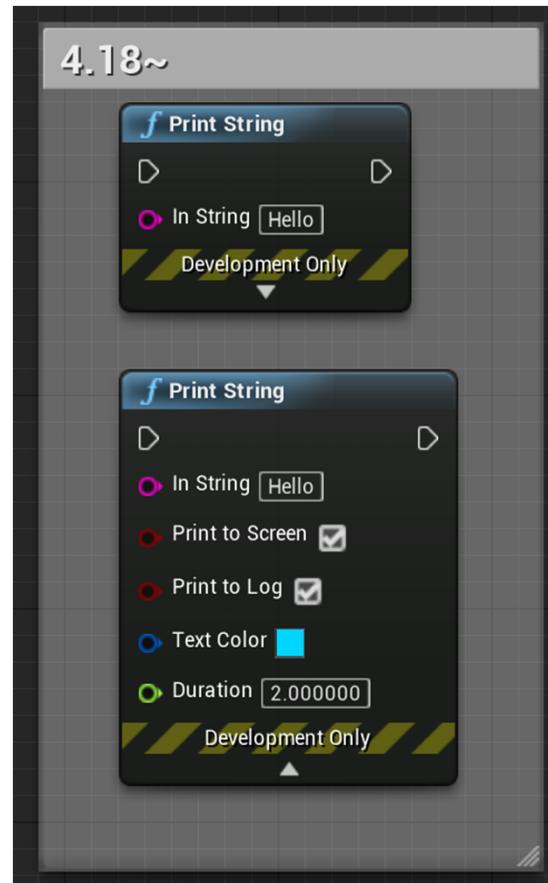
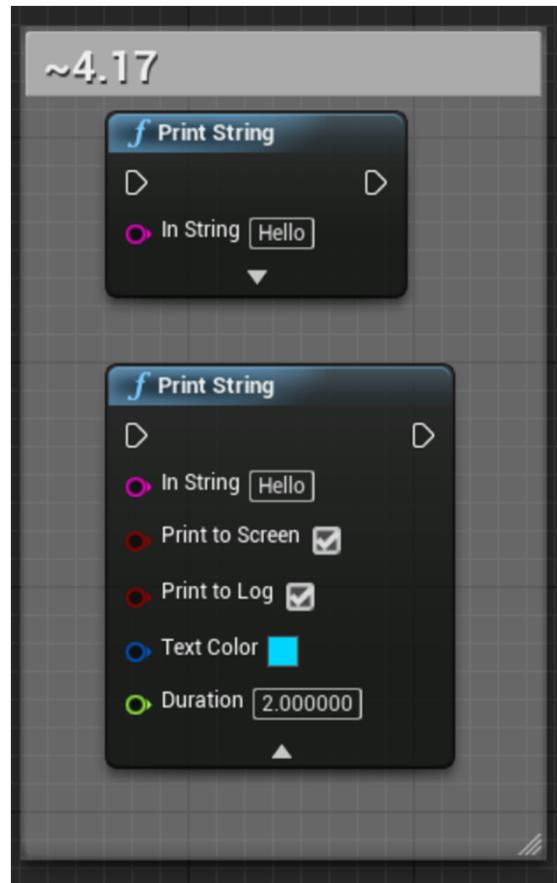
日本語翻訳

<https://github.com/akenatsu/ue4-style-guide/blob/master/README.jp.md>

UE4 Style GuideとLinterPlugin

http://denshikousakubu.com/2017/11/01/20171101_UE4StyleGuideAndLinterPlugin/

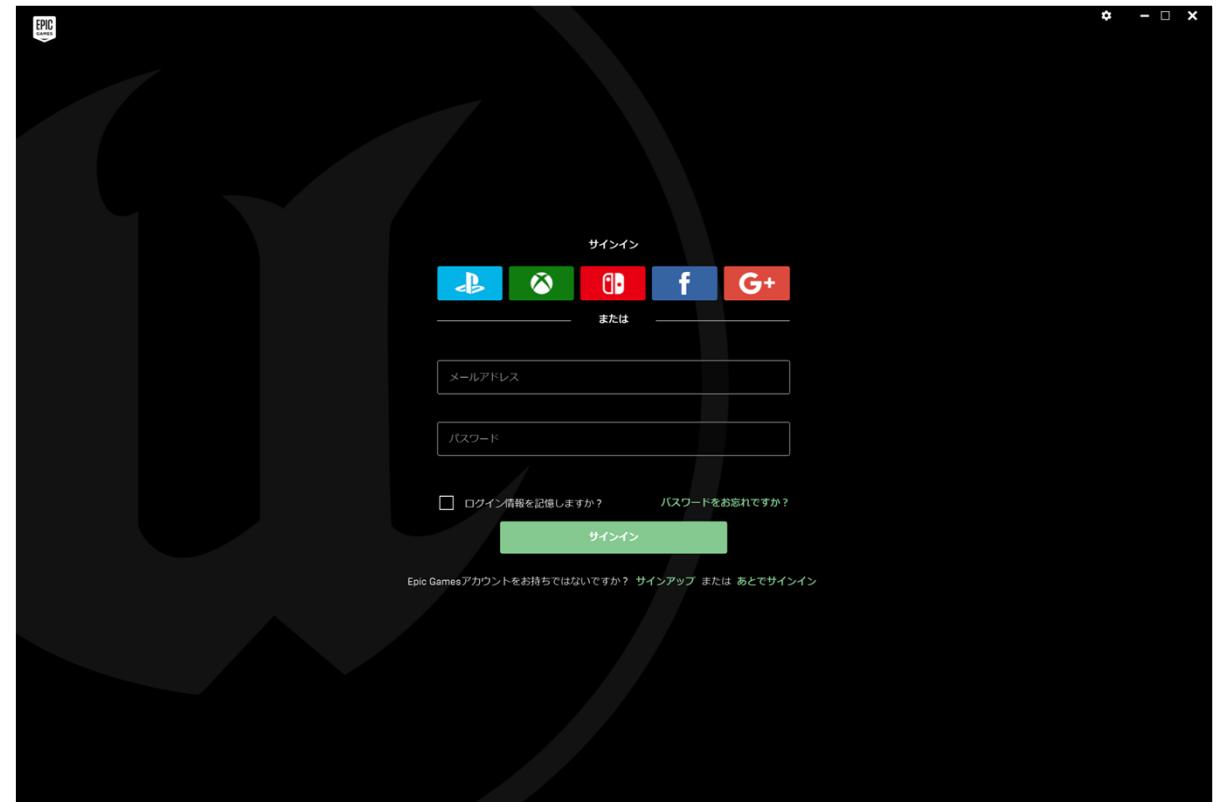
PrintStringが4.18から表示変更されました



4.17の時に作成した資料のため、
PrintStringの画像が2種類あります
4.18からPrintStringに詳細情報ボタンの上
にDevelopment Onlyの帯が表示されるよ
うになりました
PrintStringの動作は同じです

2. プロジェクトを作成する

Epic Games Launcherにログイン



Epic Games Launcherを
立ち上げる

メールアドレス・パスワードを入力してログイン

バージョンを選択して起動

1. UNREAL ENGINEを選択
2. 起動横の▼をクリック
3. 指定のバージョンを選択
[4.19.2]を選択
4. 起動ボタンをクリック



作成するプロジェクトの設定

1. 新規プロジェクトを選択
2. ブループリントタブを選択
3. 空のプロジェクトを選択
4. スターターコンテンツ無し
5. フォルダを設定
6. プロジェクト名を設定
プロジェクト名：MathGame
7. プロジェクトを作成をクリック



プロジェクトブラウザの詳細

<https://docs.unrealengine.com/latest/JPN/Engine/Basics/Projects/Browser/index.html>

フォルダの作成

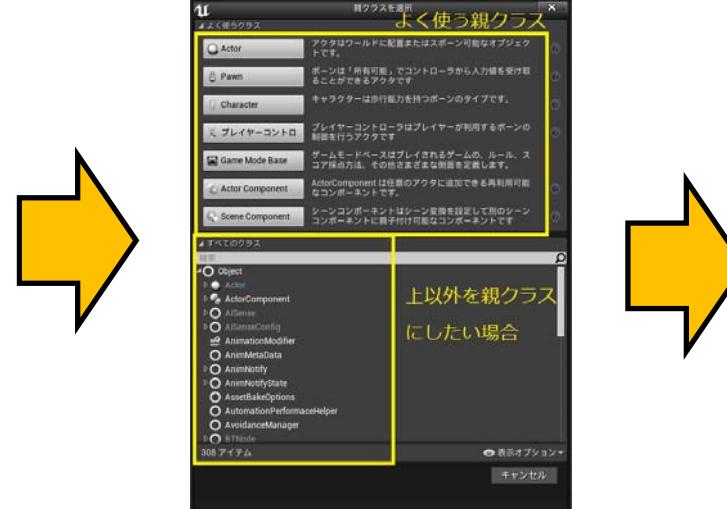
作成するフォルダ構成



3. ブループリントでHello Worldを画面に表示する

ブループリント作成の流れ

作成するフォルダ	ブループリント名	親クラス
Blueprints	BP_HelloWorld	Actor



作成するフォルダを選択
> 新規追加 or 右クリック
> ブループリントクラス

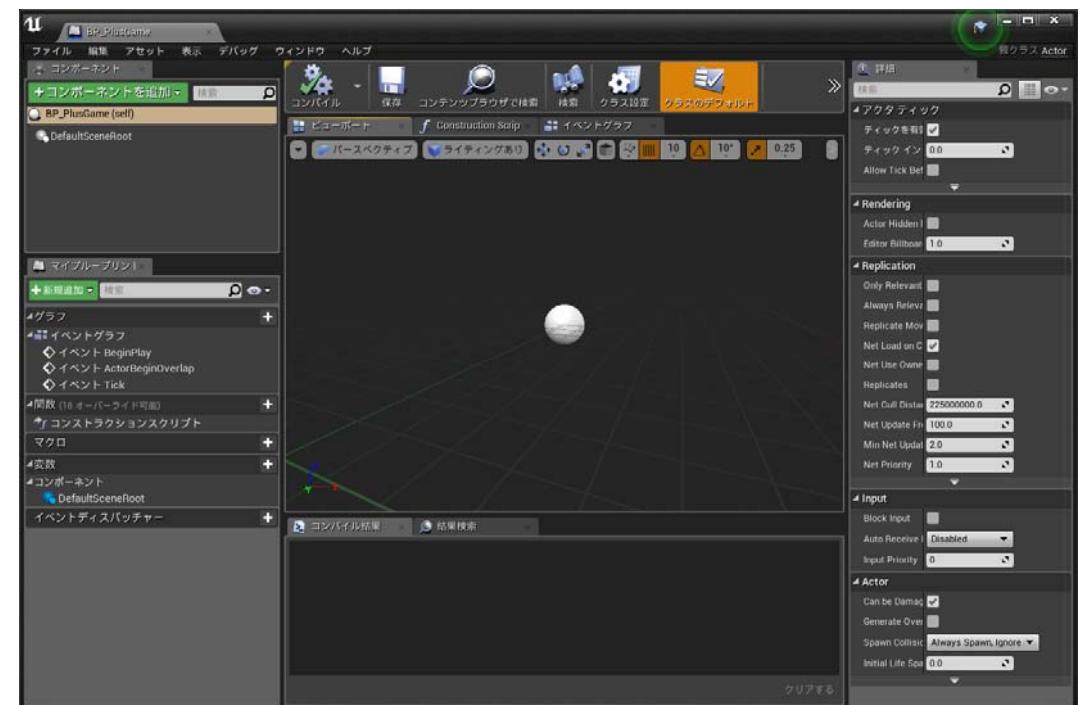
親クラスを選択
[Actor]ボタンをクリック

名前を設定する
名前を[BP_HelloWorld]に設定

ブループリントエディタを開く

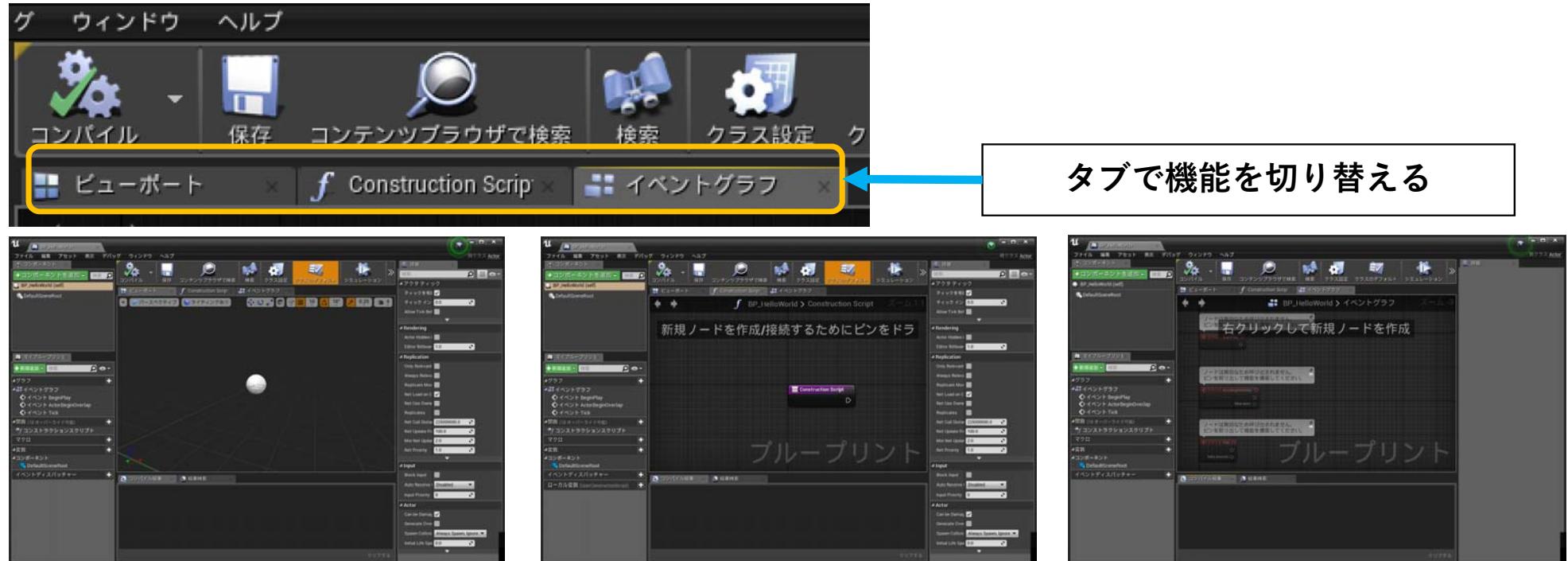


ブループリントをダブルクリック



ブループリントエディタが開く

ブループリントエディタの構成



ビューポート

コンポーネントを設定する

Construction Script

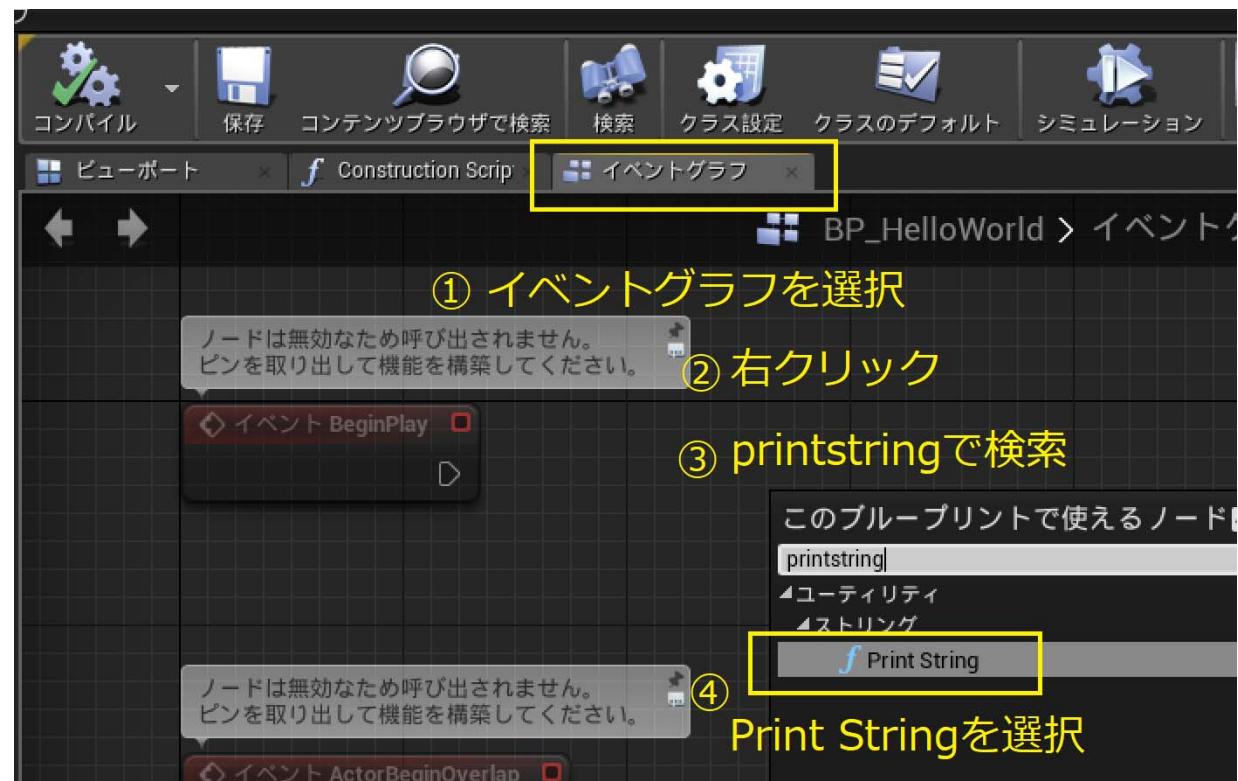
画面に配置した時に
変更できる処理を実装する

タブで機能を切り替える

イベントグラフ
処理を実装する

イベントグラフにPrint Stringを追加する

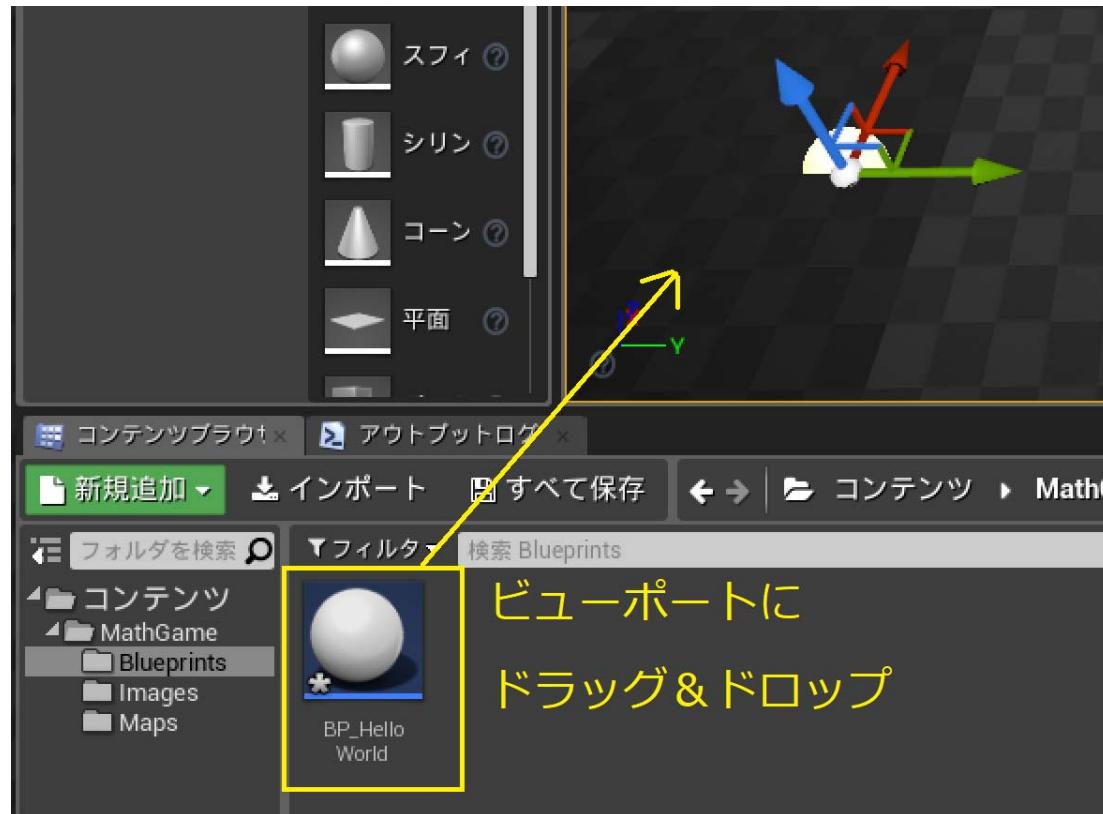
1. イベントグラフタブを選択
2. 右クリック
3. Printstringで検索
4. Print Stringを選択する



PrintStringのIn StringをHello World!に設定
イベントBeginPlayとPrintStringをつなげる



ビューポートにブループリントを
ビューポートにドラッグ&ドロップ



ビューポートに
ドラッグ&ドロップ

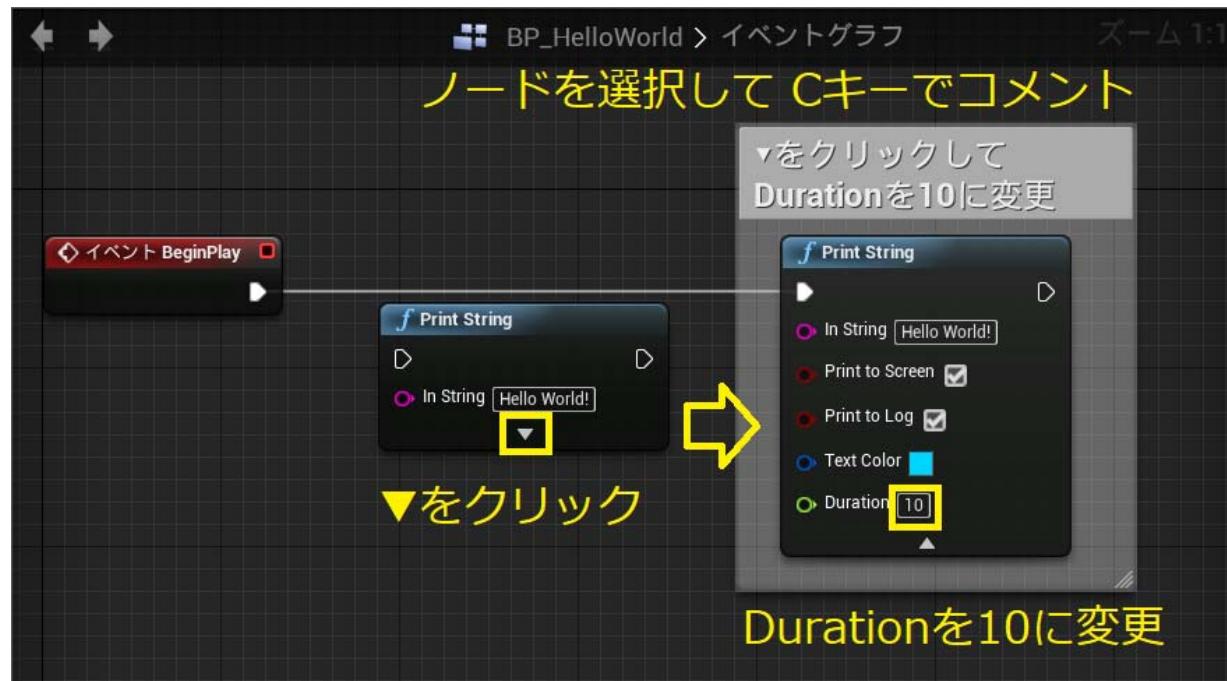
プレイをクリック
ビューポートの左上にHello World!が表示される



プレイをクリック

ビューポートの左上にPrint StringのIn Stringに
設定した文字列が表示される

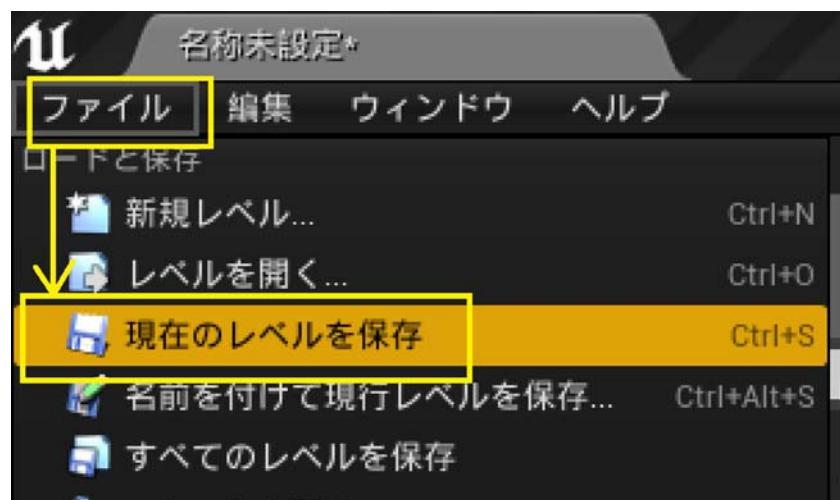
Print Stringの▼をクリック
Durationを10に設定してプレイ



▼をクリックする（ノードによってプロパティが隠されている）
Duration(表示期間：単位(秒))を10に設定する
ノードを選択してCキーでコメントを設定できる

プレイをクリック
10秒間表示される

マップを保存



ファイル > 現在のレベルを保存
ショートカットキー : Ctrl + S



Maps フォルダに HelloWorld を保存する

4. ブループリントで足し算ゲームを作成する

4. ブループリントで足し算ゲームを作成する

- 4.1 新規レベル (RandomPlusGame) を作成・保存
- 4.2 ブループリント(BP_RandomPlusGame)を作成
- 4.3 足し算の実装
- 4.4 足し算の引数を変数化
- 4.5 足し算を関数化
- 4.6 ランダムの数値を足し算する関数(RandomCalculation)を作成
- 4.7 キーボード入力で答えを入力
- 4.8 正解判定
- 4.9 算数ゲーム(足し算のみ)化

4. ブループリントで足し算ゲームを作成する

4.1 新規レベル (RandomPlusGame) を作成・保存

4.2 ブループリント(BP_RandomPlusGame)を作成

4.3 足し算の実装

4.4 足し算の引数を変数化

4.5 足し算を関数化

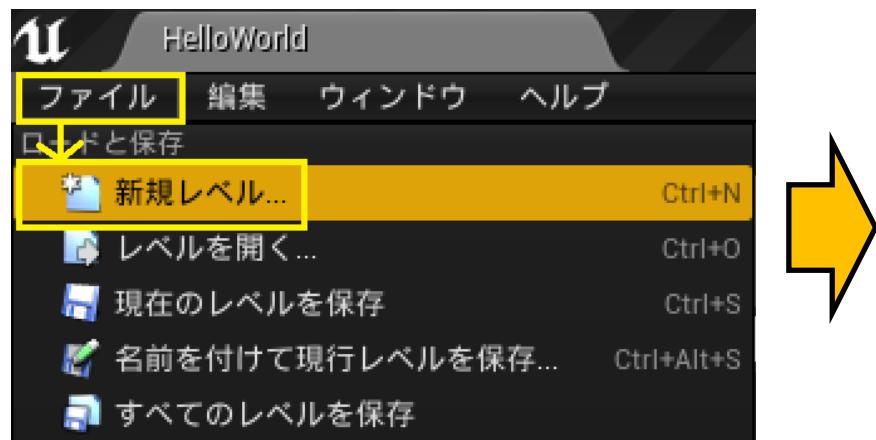
4.6 ランダムの数値を足し算する関数(RandomCalculation)を作成

4.7 キーボード入力で答えを入力

4.8 正解判定

4.9 算数ゲーム(足し算のみ)化

新規レベルの作成



ファイル > 新規レベル
ショートカット : Ctrl+N

[空のレベル]を選択する

現在のレベルを保存
名前を[RandomPlusGame]に設定して保存



ファイル > 現在のレベルを保存
ショートカット : Ctrl+S

Mapsフォルダを選択
> 名前を[RandomPlusGame]に設定
> [保存]をクリック

4. ブループリントで足し算ゲームを作成する

4.1 新規レベル (RandomPlusGame) を作成・保存

4.2 ブループリント(BP_RandomPlusGame)を作成

4.3 足し算の実装

4.4 足し算の引数を変数化

4.5 足し算を関数化

4.6 ランダムの数値を足し算する関数(RandomCalculation)を作成

4.7 キーボード入力で答えを入力

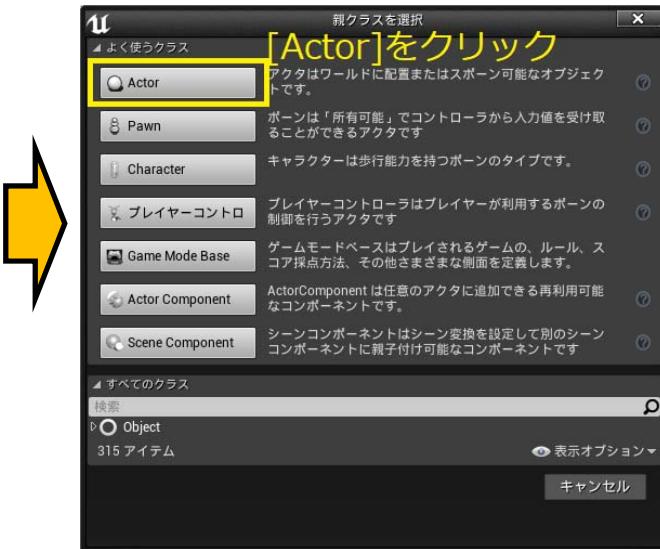
4.8 正解判定

4.9 算数ゲーム(足し算のみ)化

4.2 ブループリント (BP_RandomPlusGame)を作成

作成するフォルダ	ブループリント名	親クラス
Blueprints	BP_RandomPlusGame	Actor

Blueprints フォルダに
ブループリントを作成



親クラス > Actor をクリック



名前を [BP_RandomPlusGame] に設定

4. ブループリントで足し算ゲームを作成する

4.1 新規レベル (RandomPlusGame) を作成・保存

4.2 ブループリント(BP_RandomPlusGame)を作成

4.3 足し算の実装

4.4 足し算の引数を変数化

4.5 足し算を関数化

4.6 ランダムの数値を足し算する関数(RandomCalculation)を作成

4.7 キーボード入力で答えを入力

4.8 正解判定

4.9 算数ゲーム(足し算のみ)化

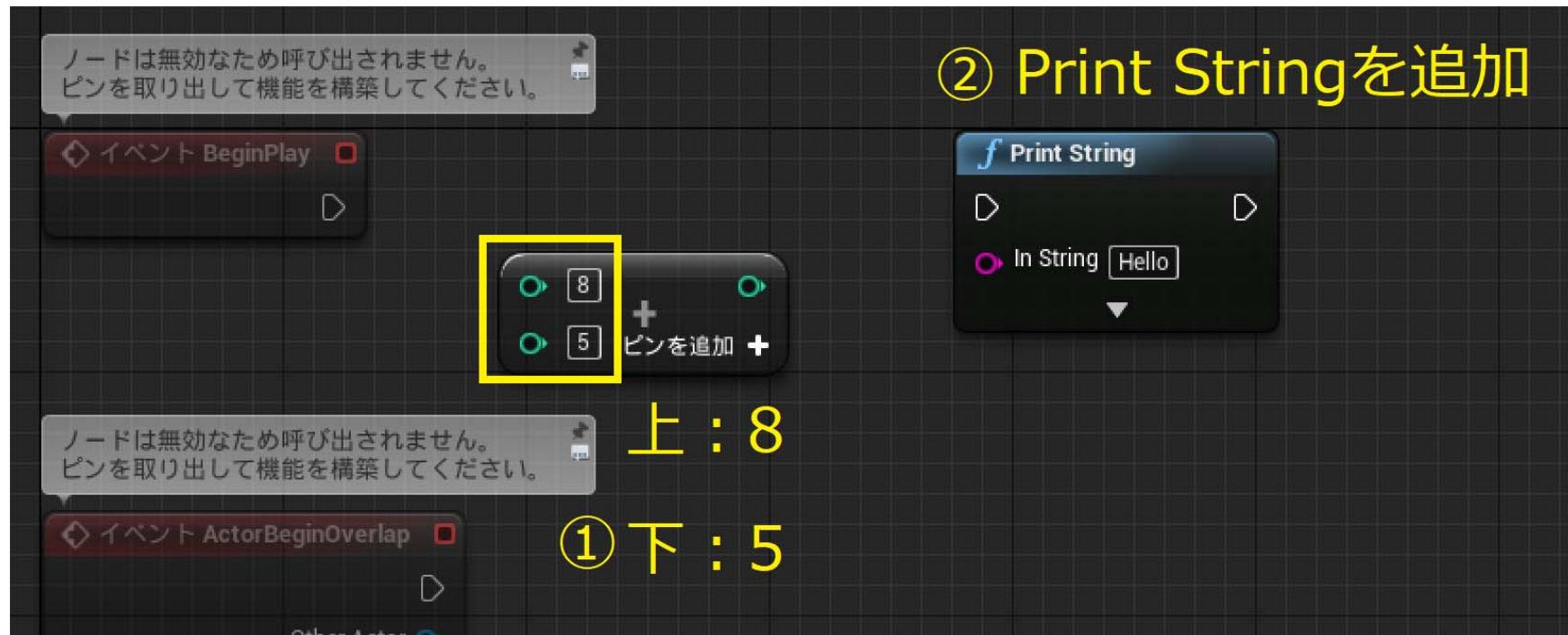
イベントグラフにinteger + integerを追加

1. イベントグラフを選択
2. 右クリック
3. 検索バーに[+]を入力
4. [integer + integer]を選択



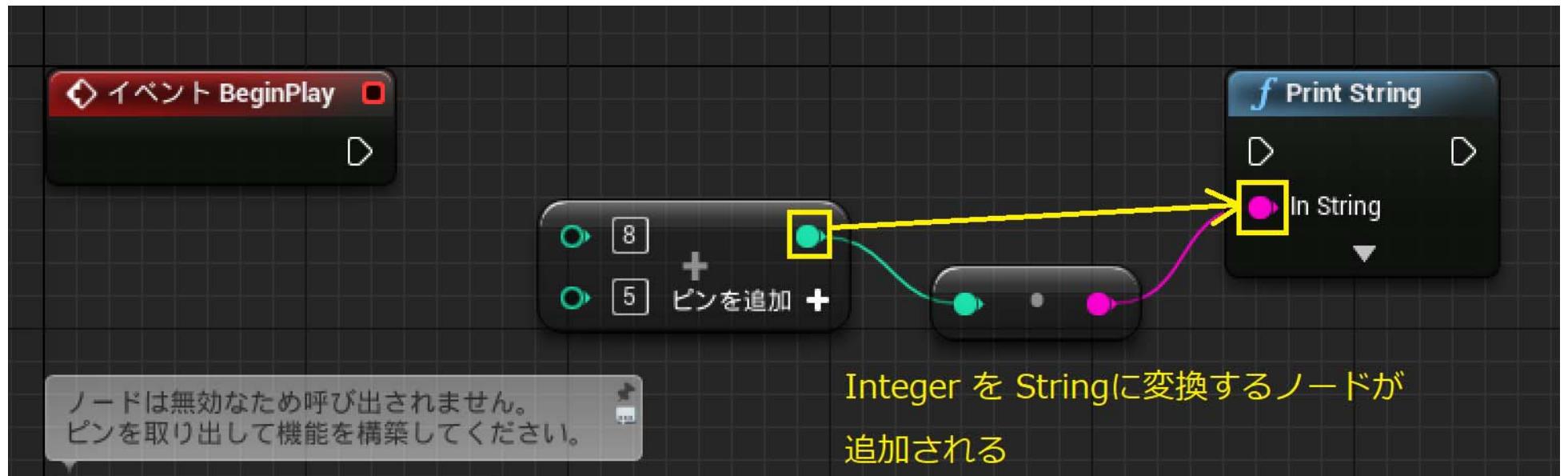
イベントグラフに足し算[integer + integer]を追加する

+ノードの上下に数値を設定
Print Stringノードを追加



1. 足し算の上下に数値を設定する
2. Print Stringを追加する

+ノードの結果をPrint StringのIn Stringにつなげる



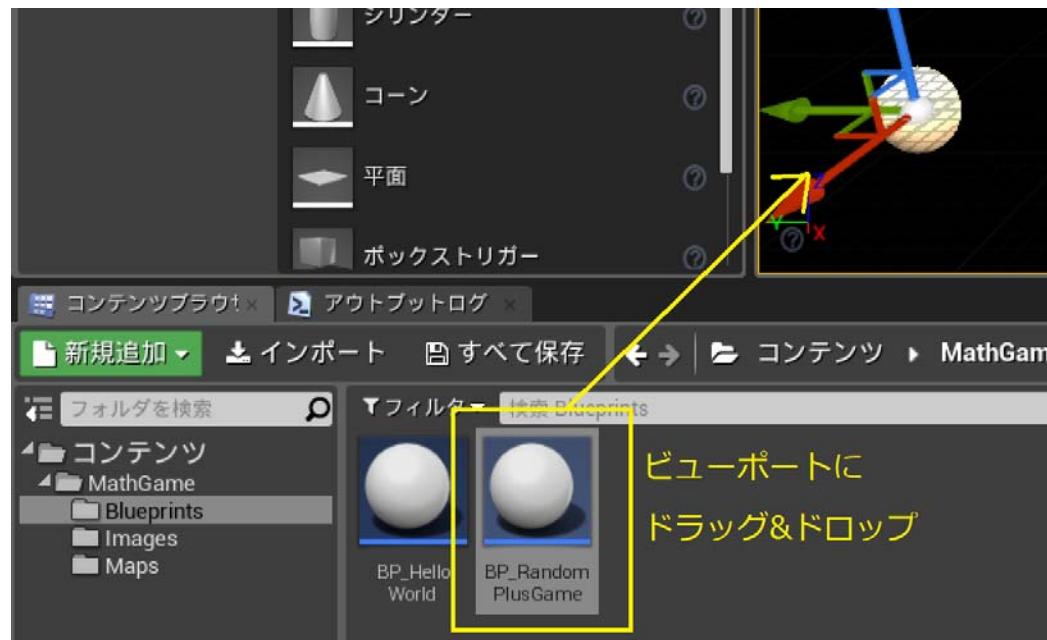
足し算ノードの結果をPrintStringのIn Stringにつなげる
整数値(Integer)が文字列(String)に変換されるノードが追加される

イベント BeginPlay と PrintString をつなげる コンパイル > 保存



1. イベント BeginPlay と PrintString をつなげる
2. コンパイル > 保存 ボタンの順でクリック
(落ちた時の為にこまめに保存は行う)

ビューポートにBP_RandomPlusGameを追加
プレイをクリックして、足し算の結果を確認



ビューポートにBP_RandomPlusGameを
ドラッグ&ドロップ



プレイをクリック
足し算の結果を確認

4. ブループリントで足し算ゲームを作成する

4.1 新規レベル (RandomPlusGame) を作成・保存

4.2 ブループリント(BP_RandomPlusGame)を作成

4.3 足し算の実装

4.4 足し算の引数を変数化

4.5 足し算を関数化

4.6 ランダムの数値を足し算する関数(RandomCalculation)を作成

4.7 キーボード入力で答えを入力

4.8 正解判定

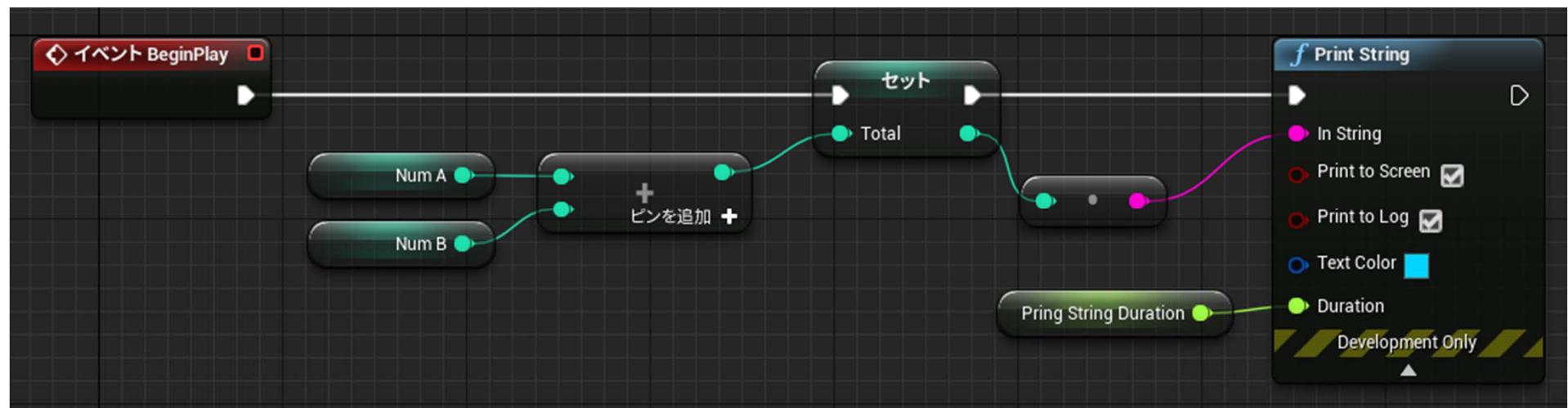
4.9 算数ゲーム(足し算のみ)化

足し算の引数を変数化

[ソースコード]

```
int NumA = 8;  
int NumB = 5;  
float PrintStringDuration = 10;  
int Total = NumA + NumB;  
PrintString(Total, PrintStringDuration);
```

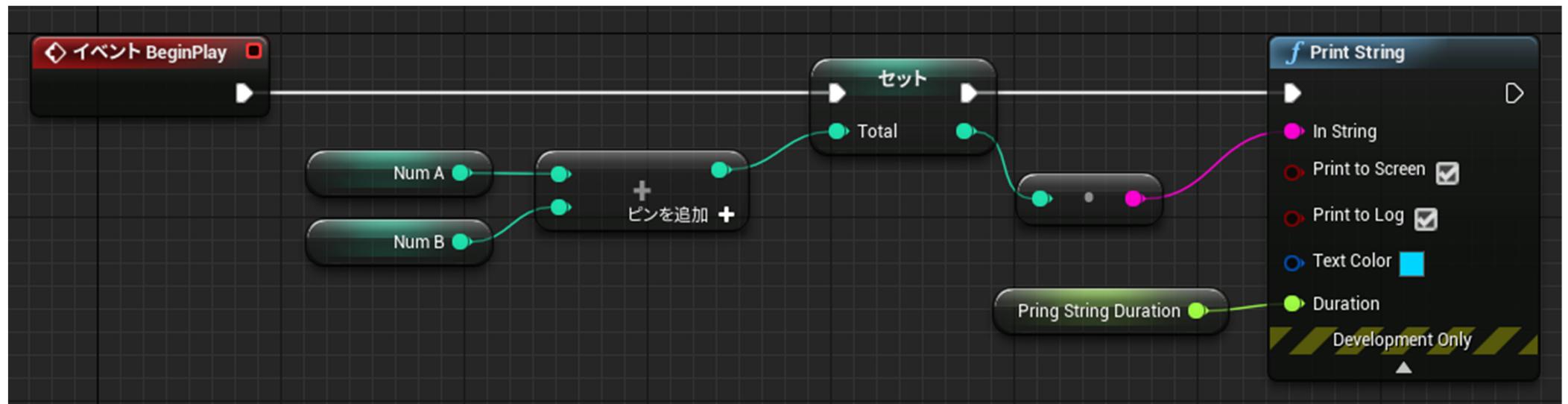
[ブループリント]



ソースコード

```
int NumA = 8;  
int NumB = 5;  
float PrintStringDuration = 10;  
int Total = NumA + NumB;  
PrintString(Total, PrintStringDuration);
```

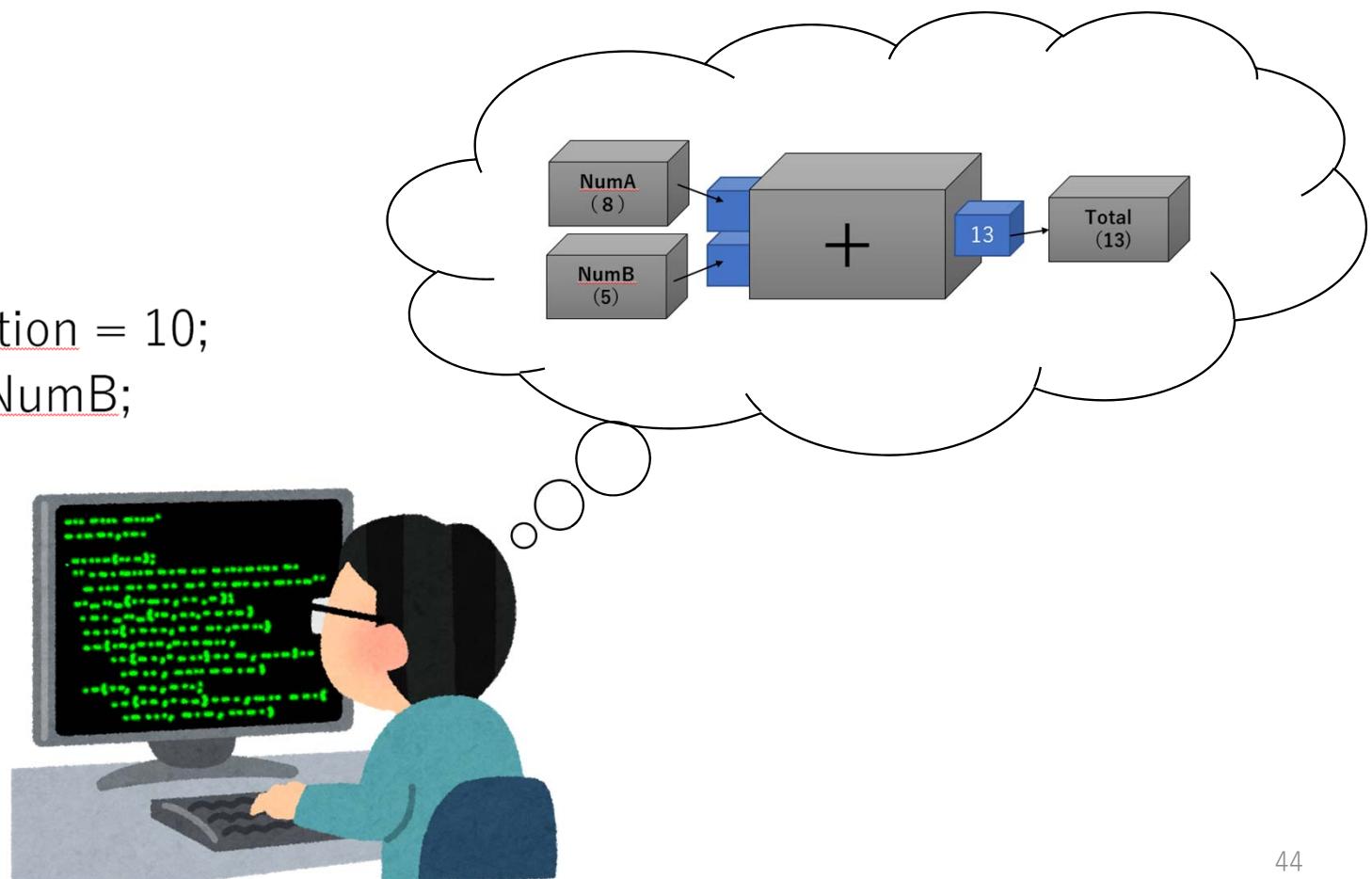
ブループリント



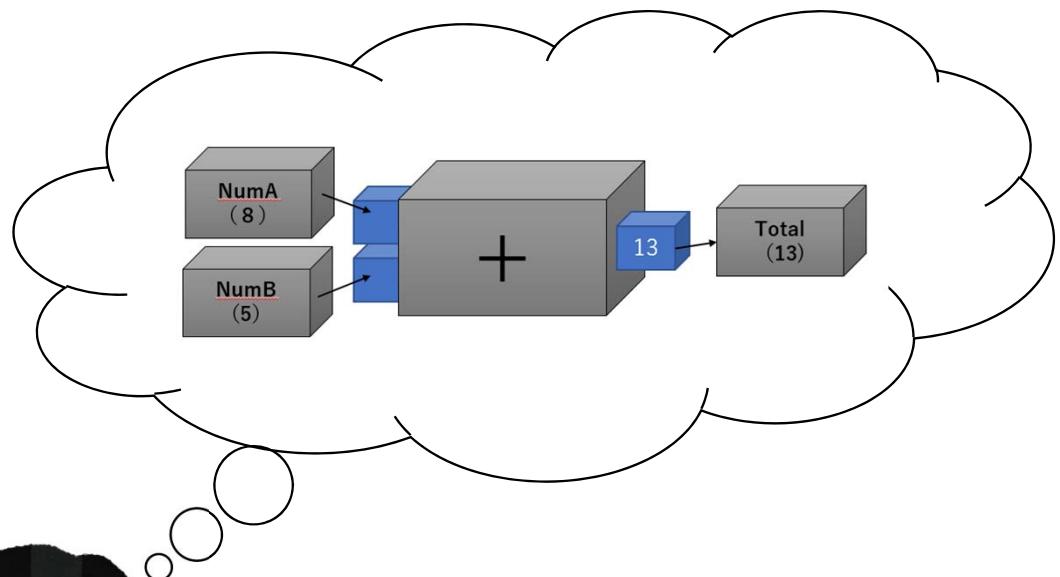
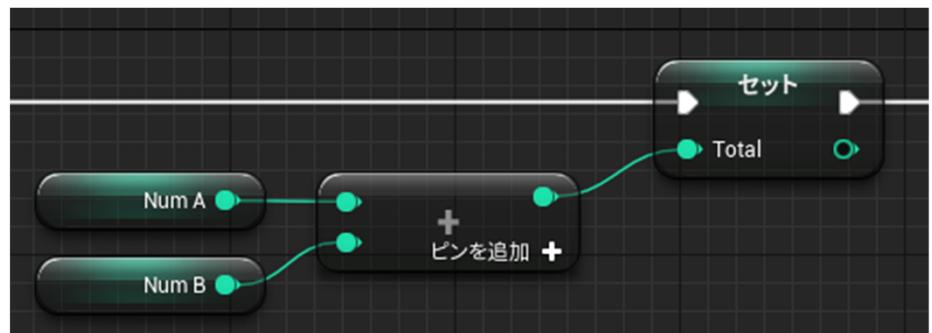
ソースコードのプログラミング

ソースコード

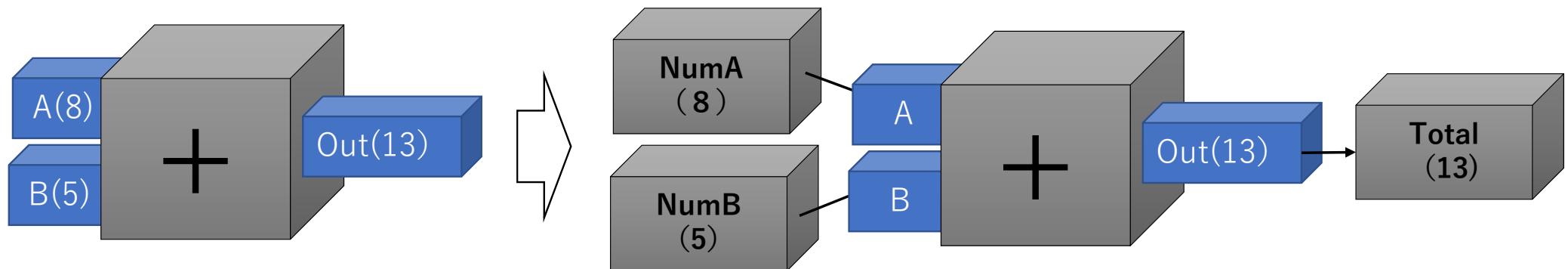
```
int NumA = 8;  
int NumB = 5;  
float PrintStringDuration = 10;  
int Total = NumA + NumB;
```



ブループリントのプログラミング



変数を使ったプログラムのイメージ



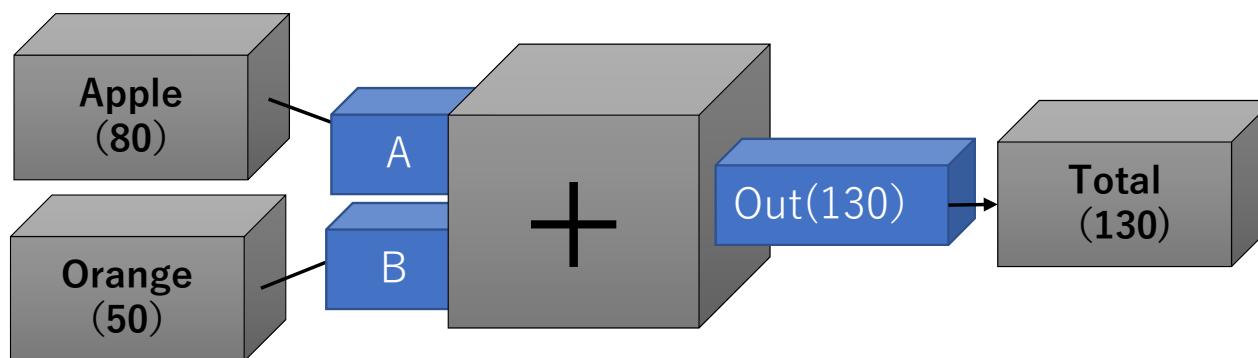
値を直接入力して使用する

8を[+]のAに入れる

変数に値を格納して、変数の名前で処理を行う

変数NumAを[+]のAに入れる

もっと分かりやすい例 果物の値段の計算



変数の追加し、変数の型をIntegerに設定

変数名	変数の型	デフォルト値
NumA	Integer	8



名前を[NumA]に設定]



変数の型をIntegerに設定

デフォルト値を設定する

変数名	変数の型	デフォルト値
NumA	Integer	8



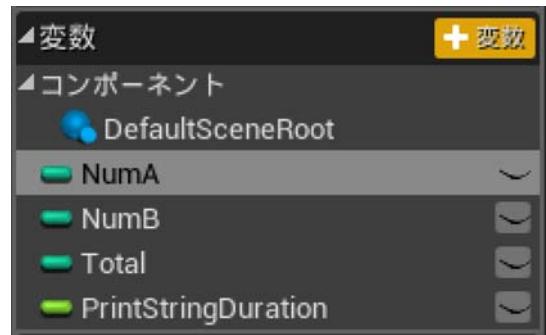
[コンパイル]をクリック
(コンパイルしないと
デフォルト値が設定できない)



デフォルト値に[8]を設定する

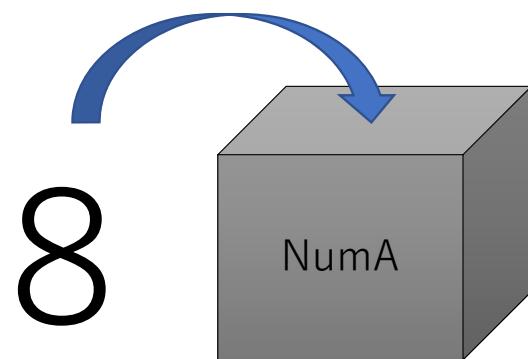
ソースコードで書くと
int NumA = 8;

使用する変数を設定する



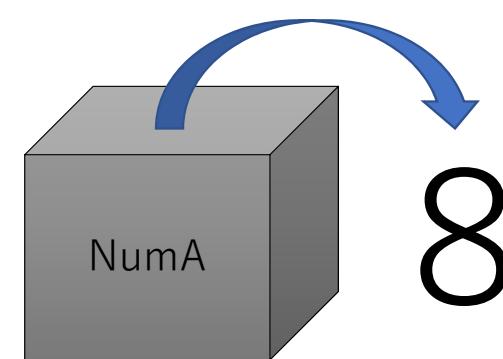
変数名	変数の型	デフォルト値
NumA	Integer	8
NumB	Integer	5
Total	Integer	-(設定なし)
PrintStringDuration	Float	10.0

変数のGet/Setのイメージ



Set

変数に値を格納する



Get

変数から値を取り出す

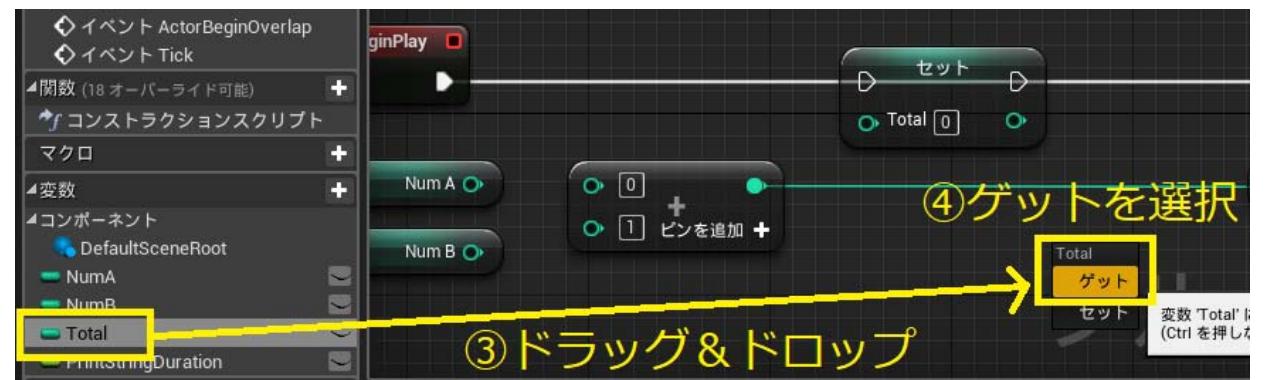
変数(NumA,NumB)のゲットを追加する

1. NumAをドラッグ & ドロップ
2. ゲットを選択
3. NumBをドラッグ & ドロップ
4. ゲットを選択



変数Totalのセットとゲットを追加

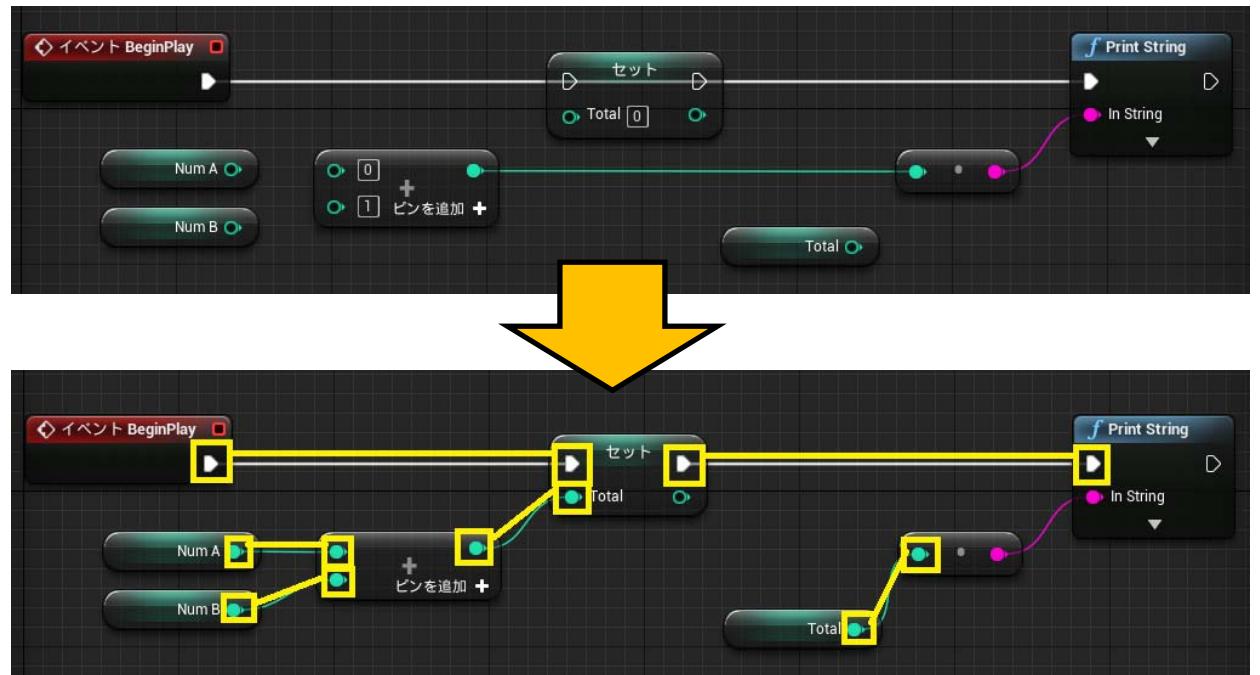
1. Totalをドラッグ & ドロップ
2. セットを選択
3. Totalをドラッグ & ドロップ
4. ゲットを選択



ノードをつなげる

```
PrintString((0+1).ToString());
```

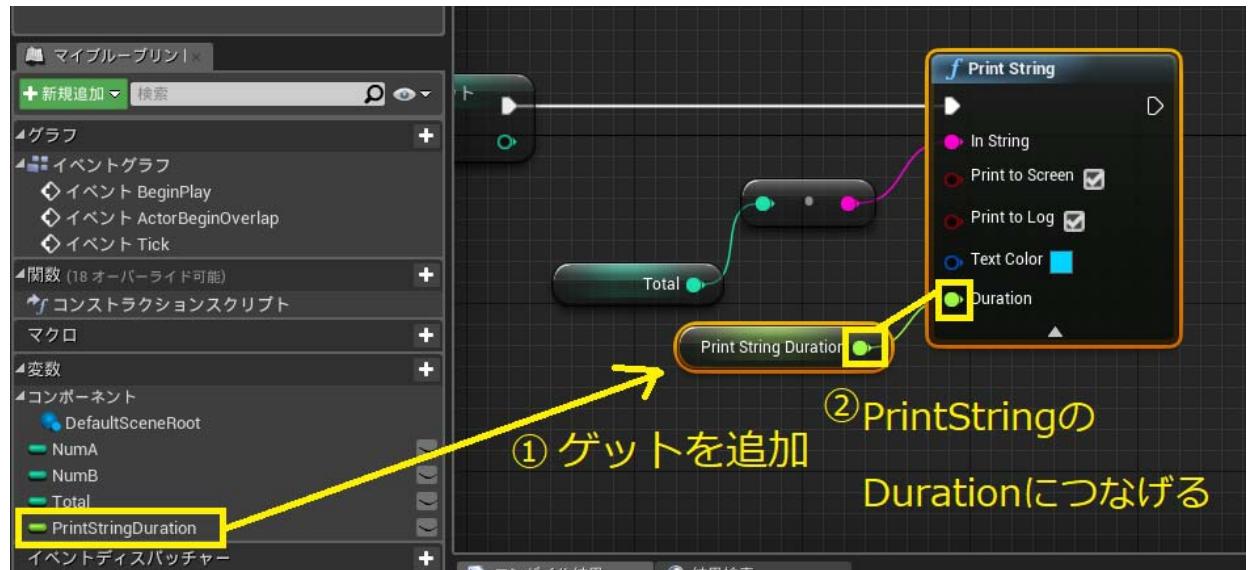
```
int NumA = 8;  
int NumB = 5;  
int Total = NumA + NumB;  
PrintString(Total.ToString());
```



PrintStringDurationのゲットを追加し、PrintStringのDurationにつなげる

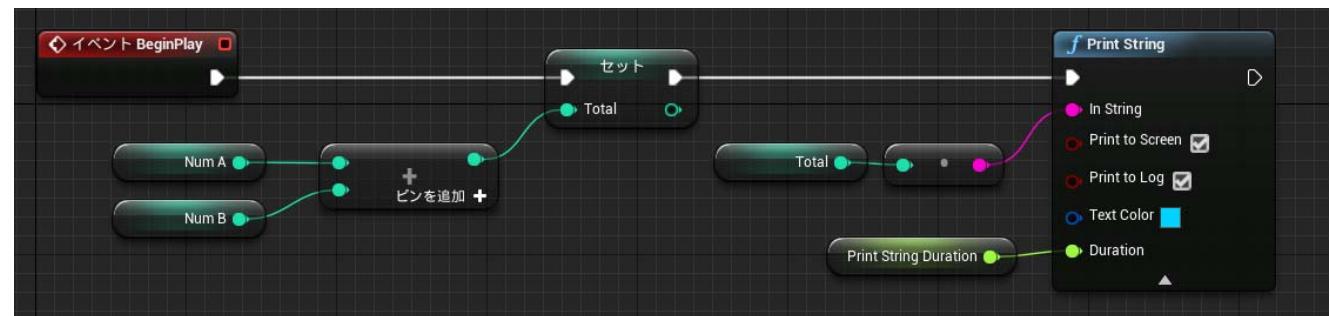
1. PrintStringDurationのゲットを追加する
2. PrintStringDurationのゲットをPrintStringのDurationにつなげる

```
int NumA = 8;  
int NumB = 5;  
float PrintStringDuration =10;  
int Total = NumA + NumB;  
PrintString(Total.ToString(), PrintStringDuration);
```

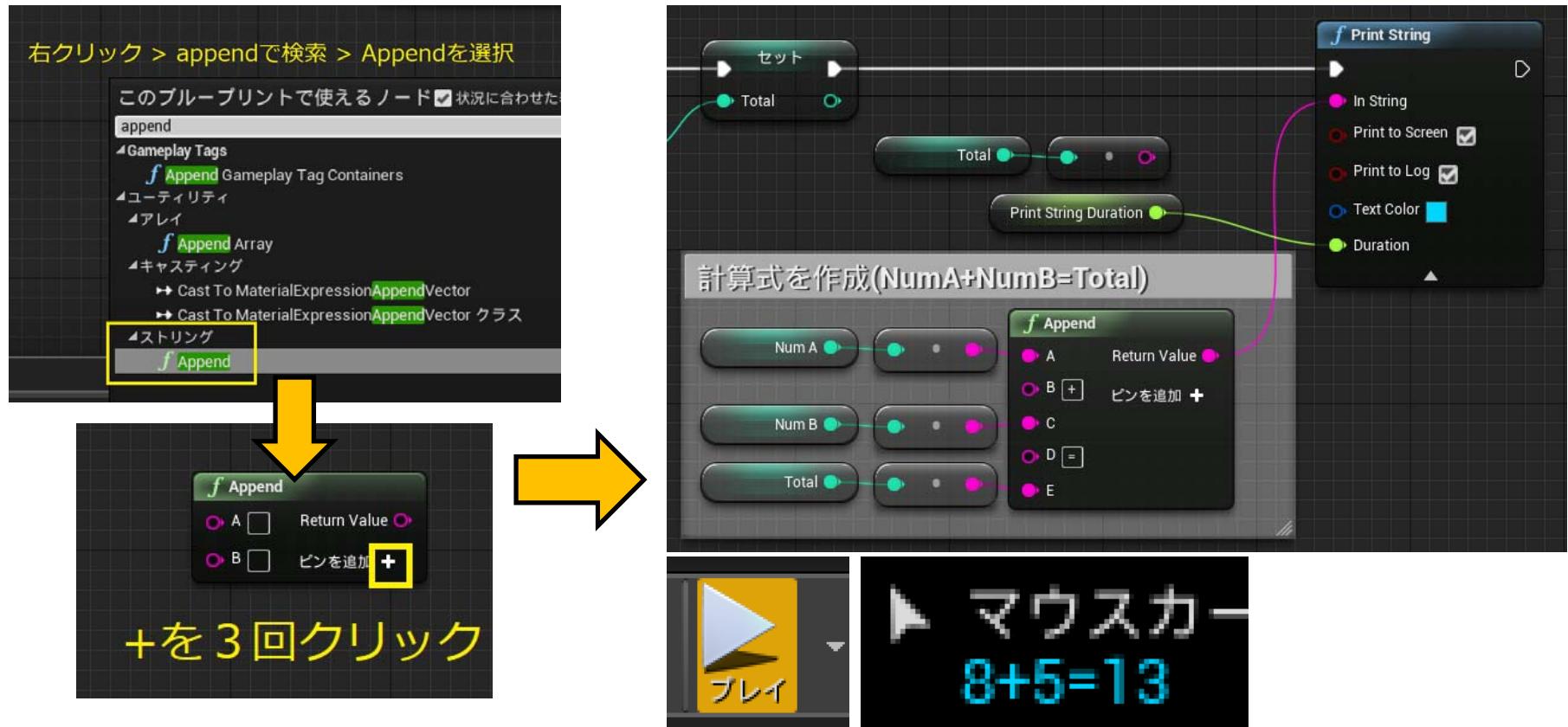


プレイして確認

```
int NumA = 8;  
int NumB = 5;  
float PrintStringDuration = 10;  
int Total = NumA + NumB;  
PrintString(Total.ToString(), PrintStringDuration);
```



文字列を連結するAppendノードで計算式を表示する



4. ブループリントで足し算ゲームを作成する

4.1 新規レベル (RandomPlusGame) を作成・保存

4.2 ブループリント(BP_RandomPlusGame)を作成

4.3 足し算の実装

4.4 足し算の引数を変数化

4.5 足し算を関数化

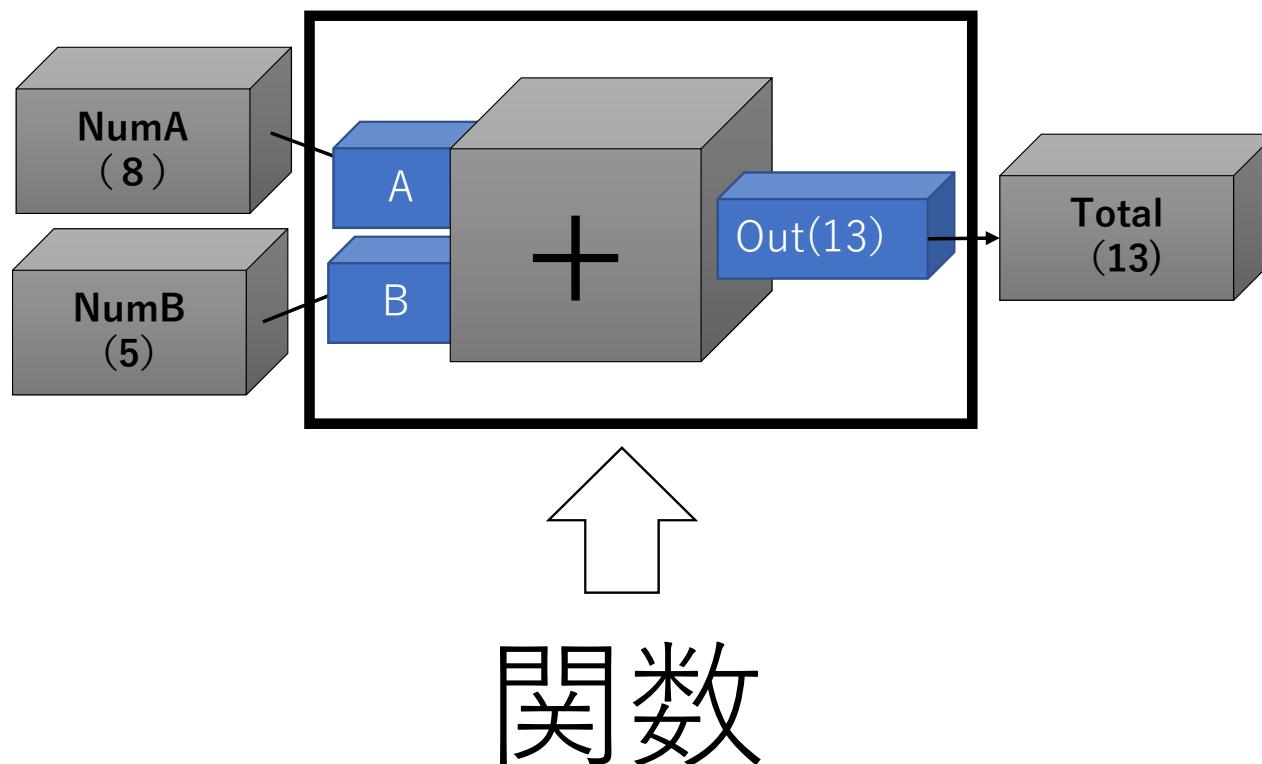
4.6 ランダムの数値を足し算する関数(RandomCalculation)を作成

4.7 キーボード入力で答えを入力

4.8 正解判定

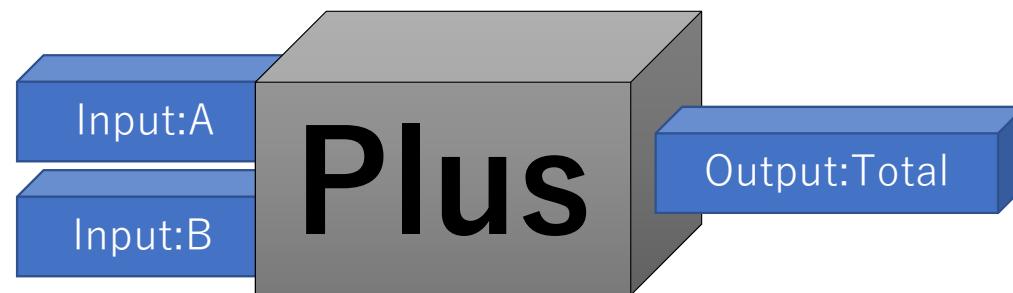
4.9 算数ゲーム(足し算のみ)化

足し算をする関数を自作する



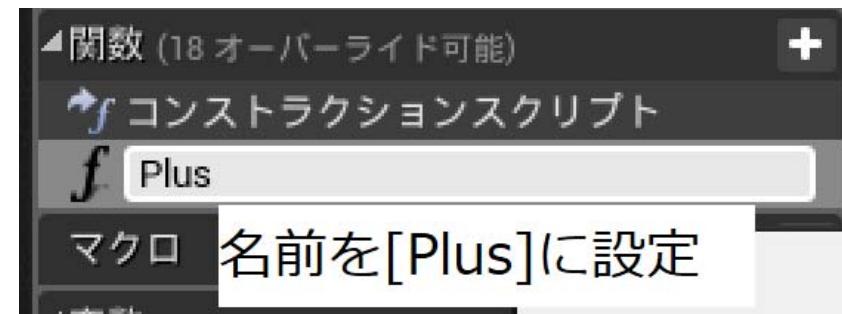
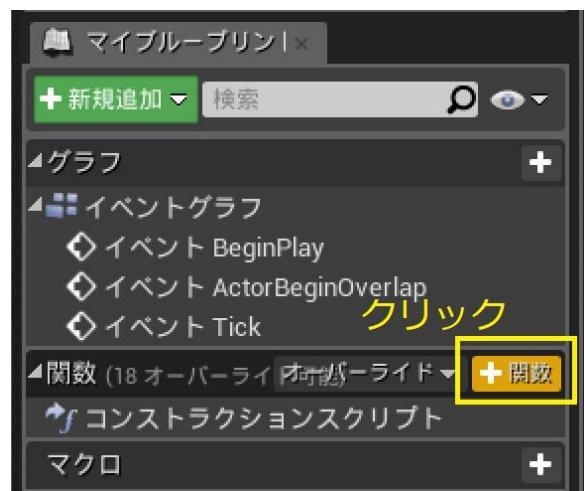
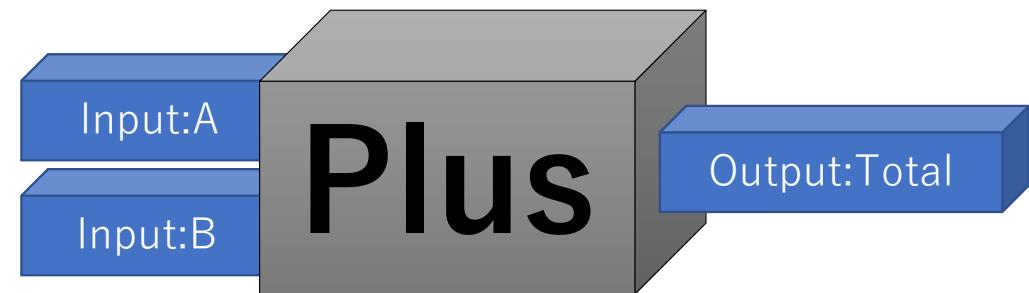
関数 Plus の作成

関数名	Input/Output	パラメータ名	型
Plus	Input	A	Integer
	Input	B	Integer
	Output	Total	Integer



関数 Plus の作成

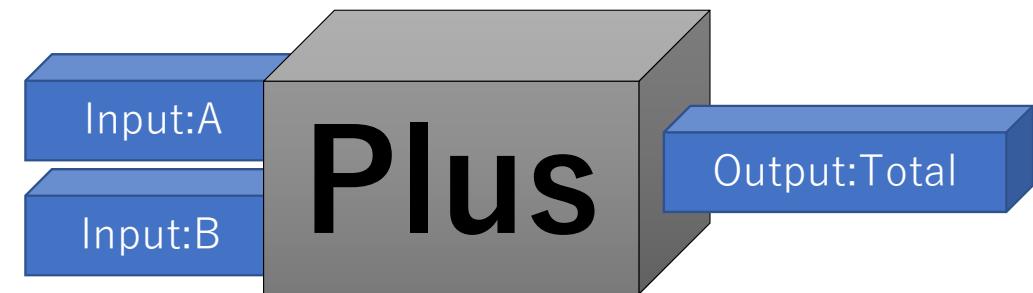
関数名	Input/Output	パラメータ名	型
Plus	Input	A	Integer
	Input	B	Integer
	Output	Total	Integer



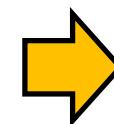
マイブループリントタブ
[+関数]をクリック

インプット、アウトプットのパラメータ設定

関数名	Input/Output	パラメータ名	型
Plus	Input	A	Integer
	Input	B	Integer
	Output	Total	Integer



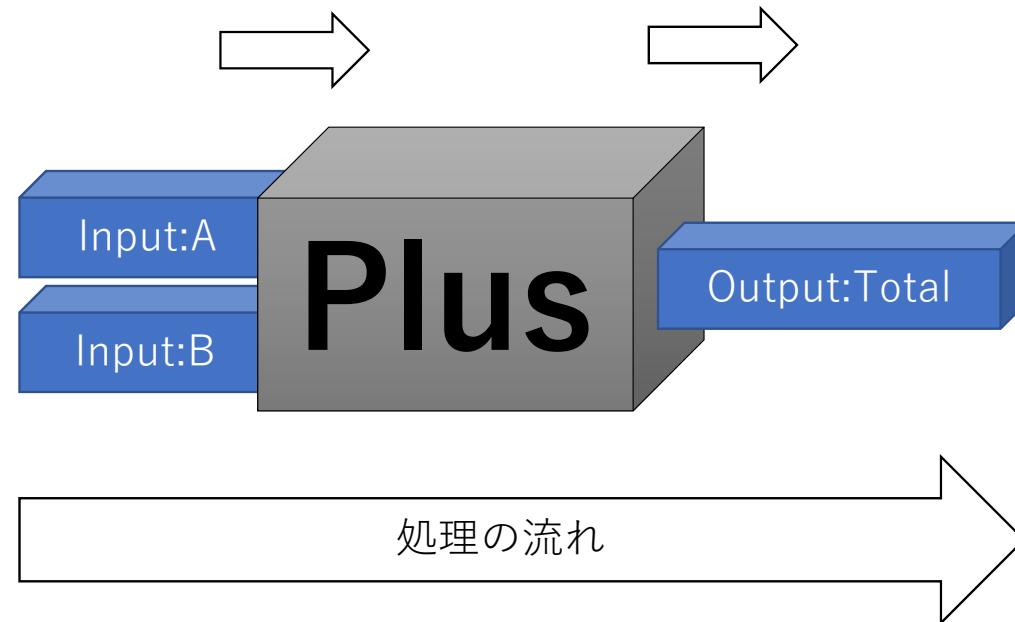
インプットを設定



アウトプットを設定

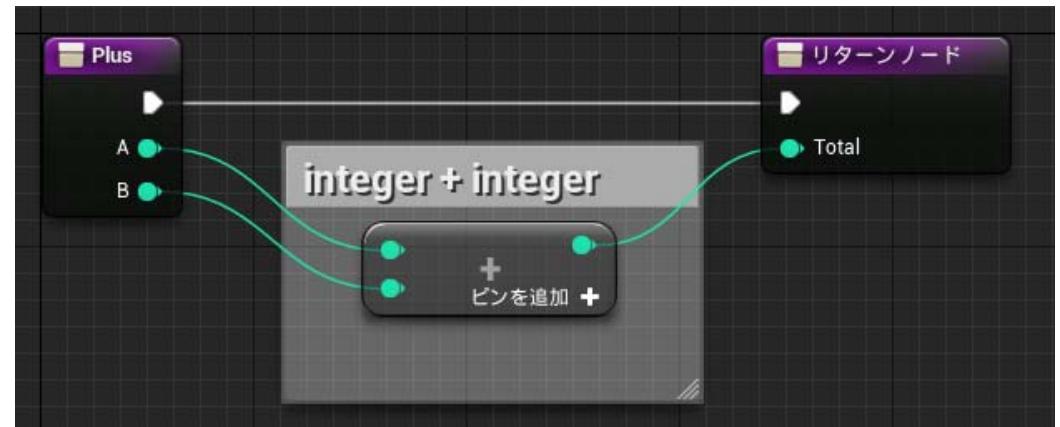
インプット、アウトプットのイメージ

中に入るのでインプット 外に出るのでアウトプット

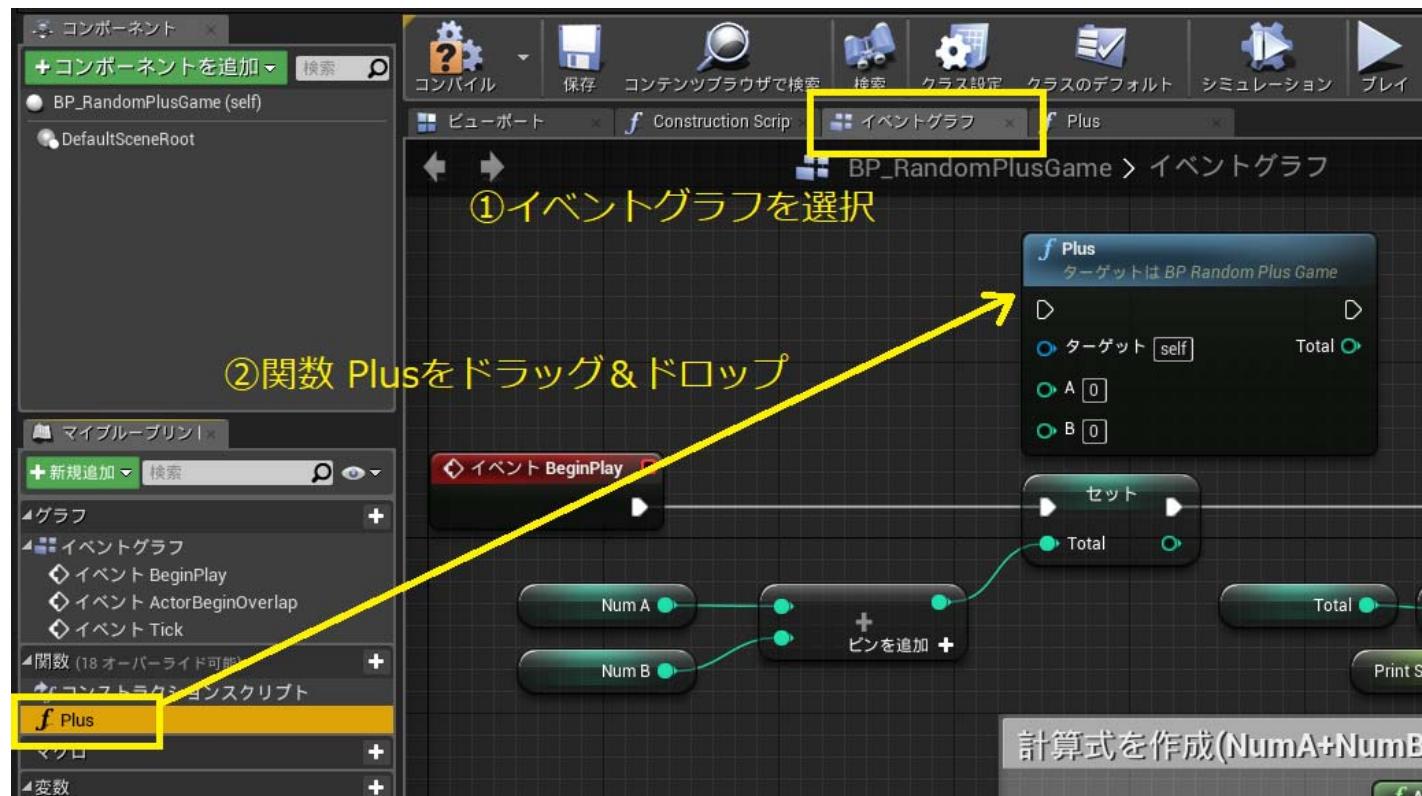


処理を実装

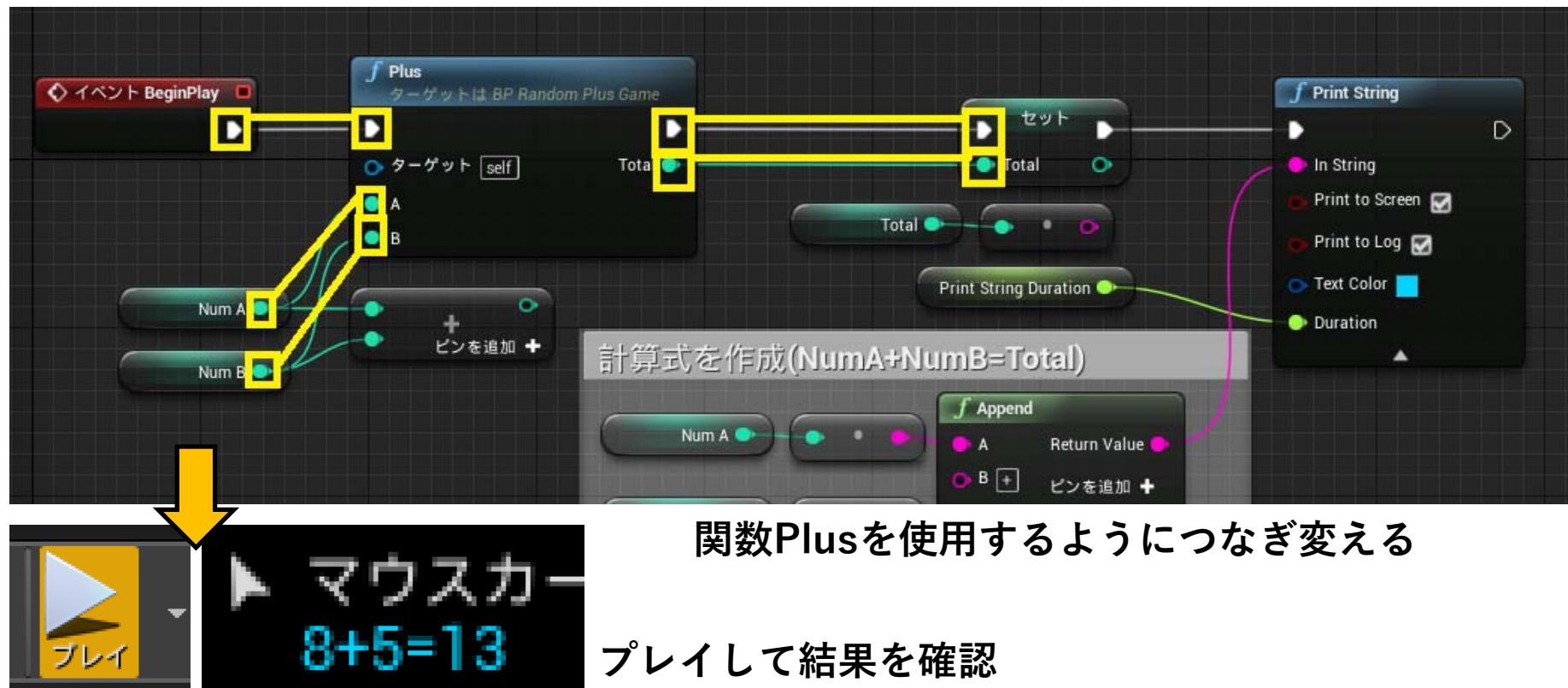
```
int Plus(int A, int B)  
{  
    return A + B;  
}
```



関数 Plus をドラッグ & ドロップ



関数 Plusを使用するようにつなぎ変える



4. ブループリントで足し算ゲームを作成する

4.1 新規レベル (RandomPlusGame) を作成・保存

4.2 ブループリント(BP_RandomPlusGame)を作成

4.3 足し算の実装

4.4 足し算の引数を変数化

4.5 足し算を関数化

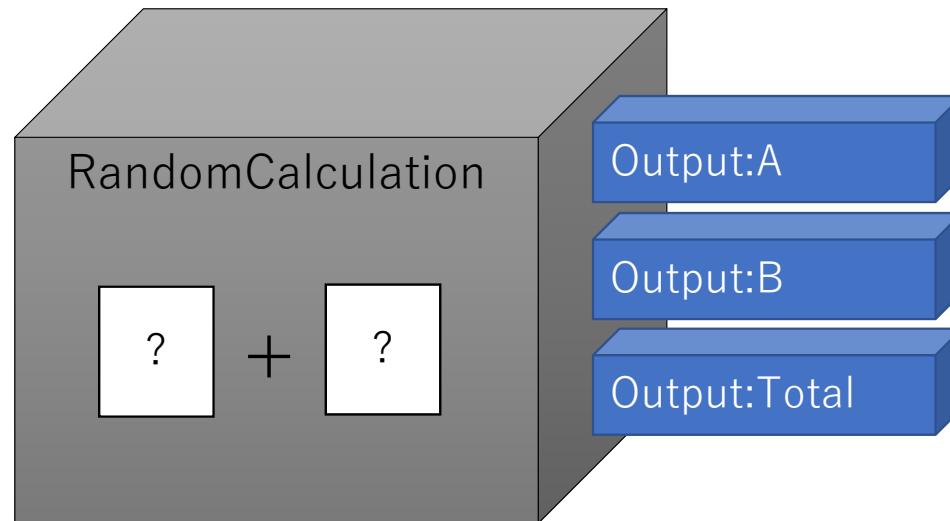
4.6 ランダムの数値を足し算する関数(RandomCalculation)を作成

4.7 キーボード入力で答えを入力

4.8 正解判定

4.9 算数ゲーム(足し算のみ)化

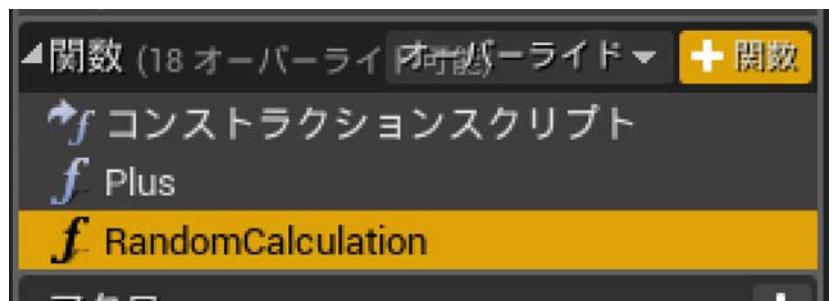
ランダムの数値を足し算する関数 (RandomCalculation)



ランダムの数値A,Bを足し算する関数

関数:RandomCalculationの作成

関数名	Input/Output	パラメータ名	型
RandomCalculation	Output	A	Integer
	Output	B	Integer
	Output	Total	Integer



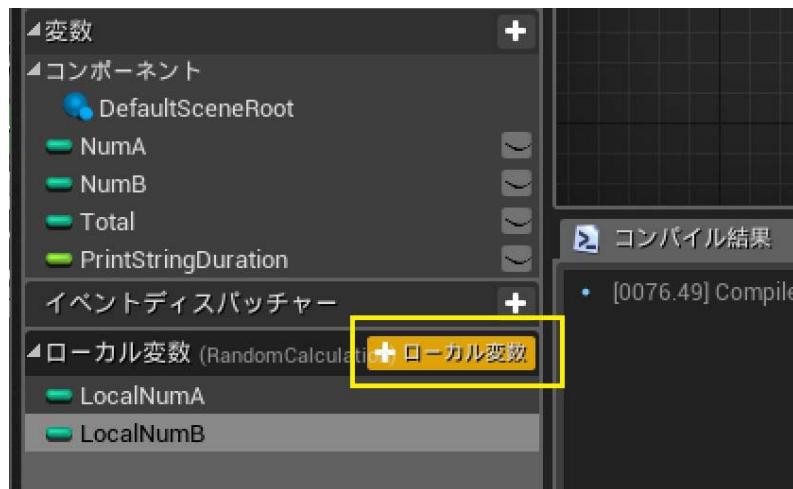
[+関数]をクリック
名前を[RandomCalculation]に設定



アウトプットのパラメータを設定する

ローカル変数を追加する

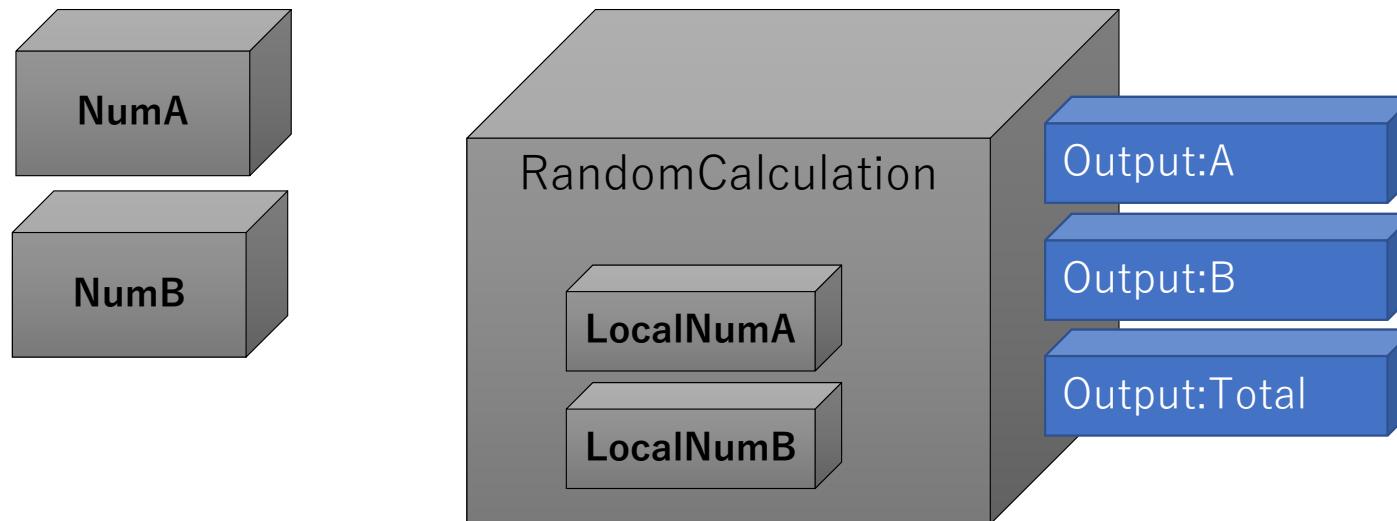
ローカル変数名	変数の型
LocalNumA	Integer
LocalNumB	Integer



ローカル変数を追加する
(ローカル変数は関数内のみ有効)

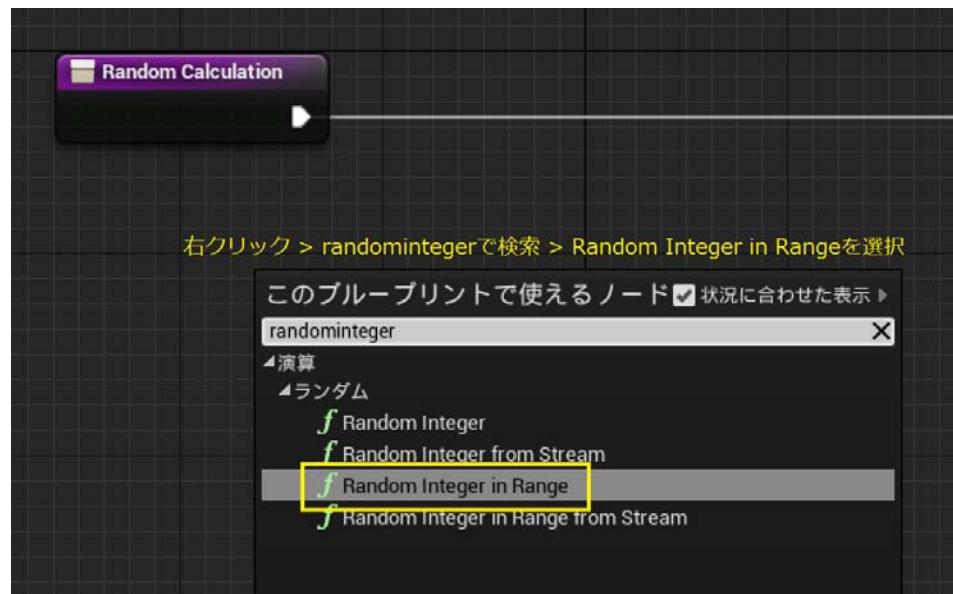
変数とローカル変数

BP_RandomPlusGame

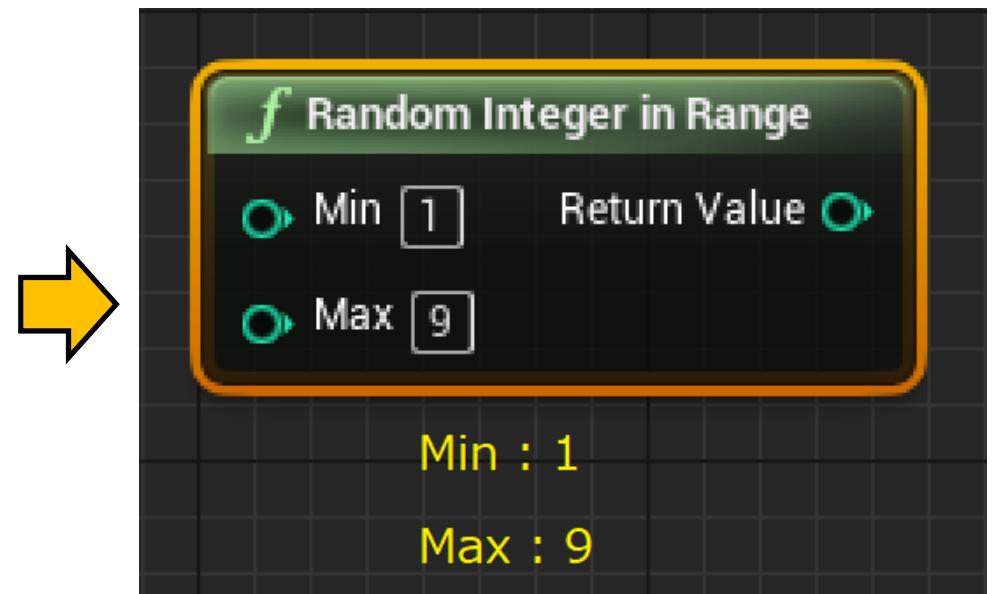


変数	ブループリントクラス内どこでも使用することが出来る
ローカル変数	関数内でしか使用することができない

指定した範囲内のRandom値を返す Random Integer in Rangeを追加・設定

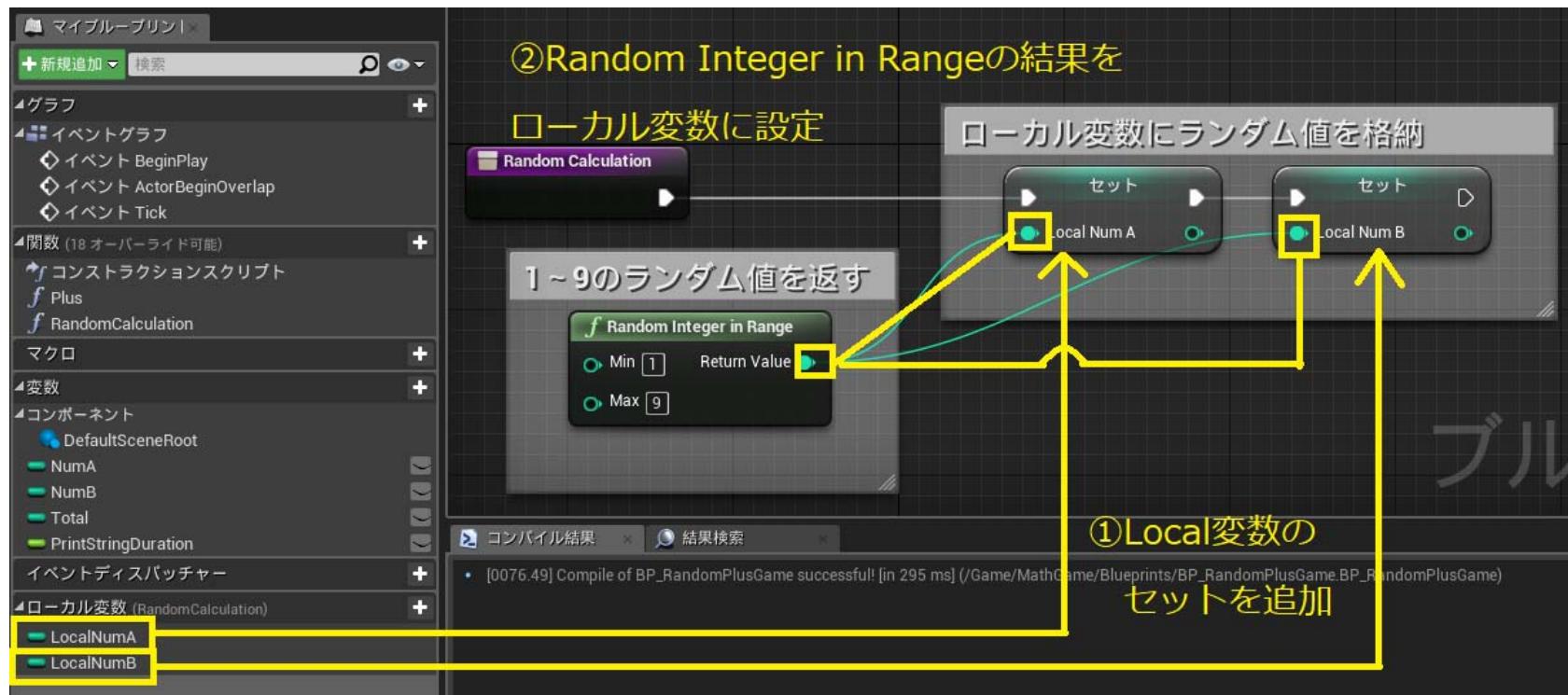


右クリック > randomintegerで検索
> Random Integer in Rangeを選択

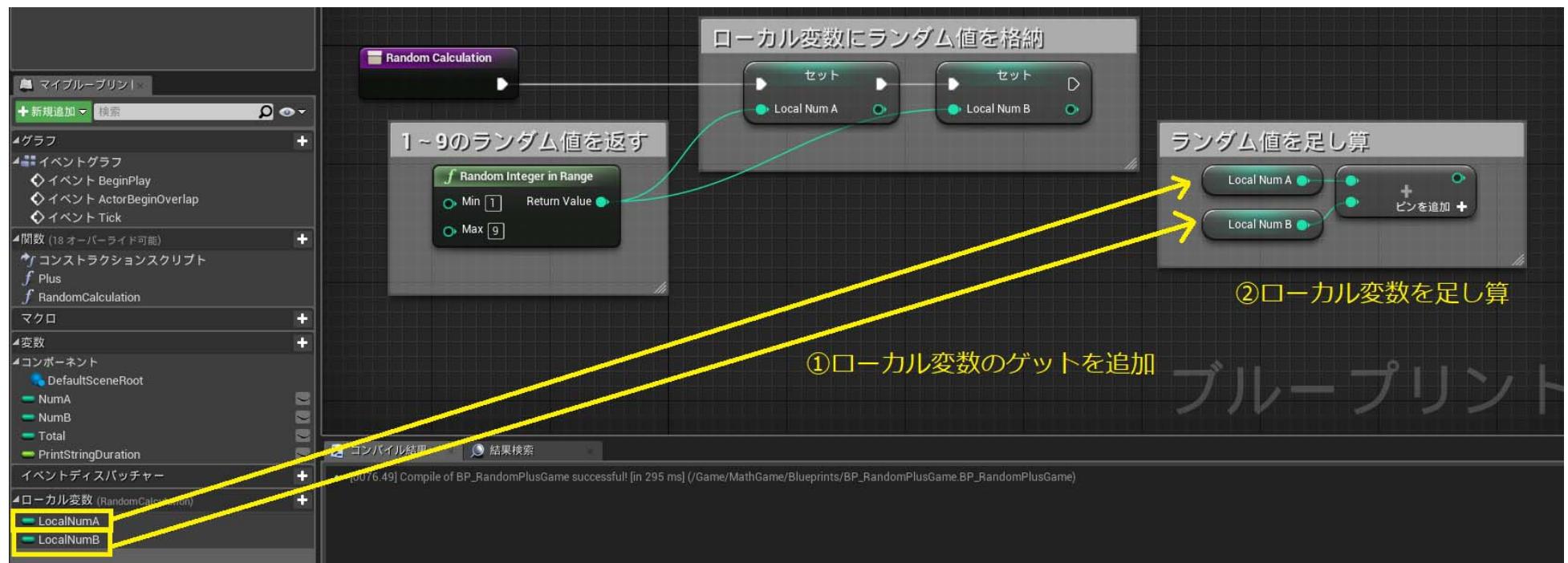


1~9のランダム値を返すように
Min,Maxの値を設定

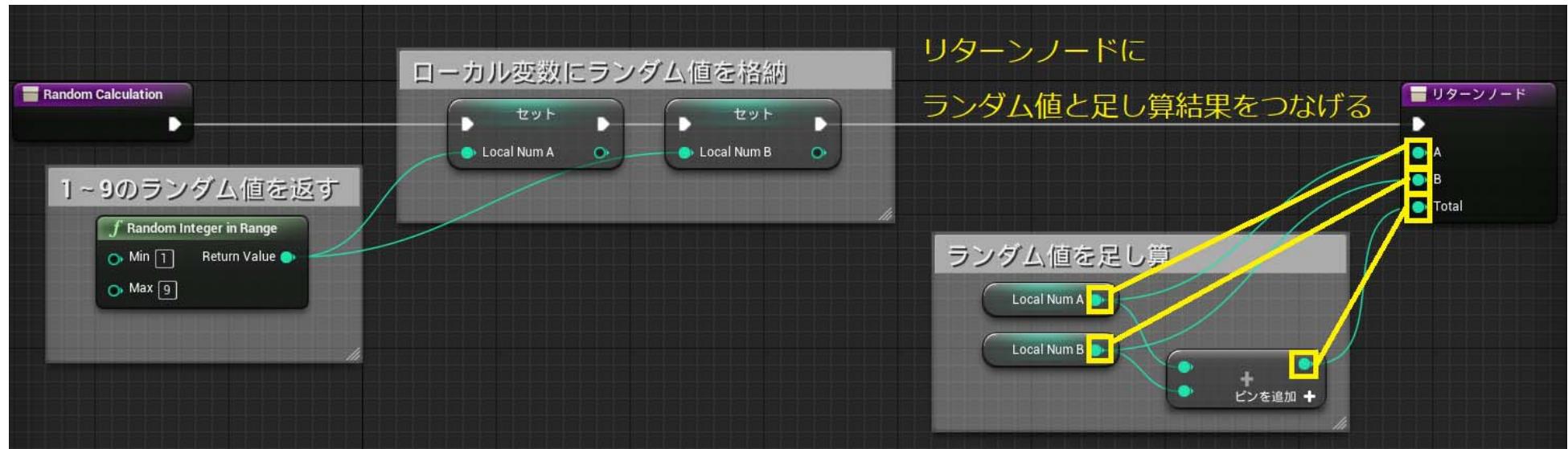
ローカル変数に Random Integer in Rangeの値を設定



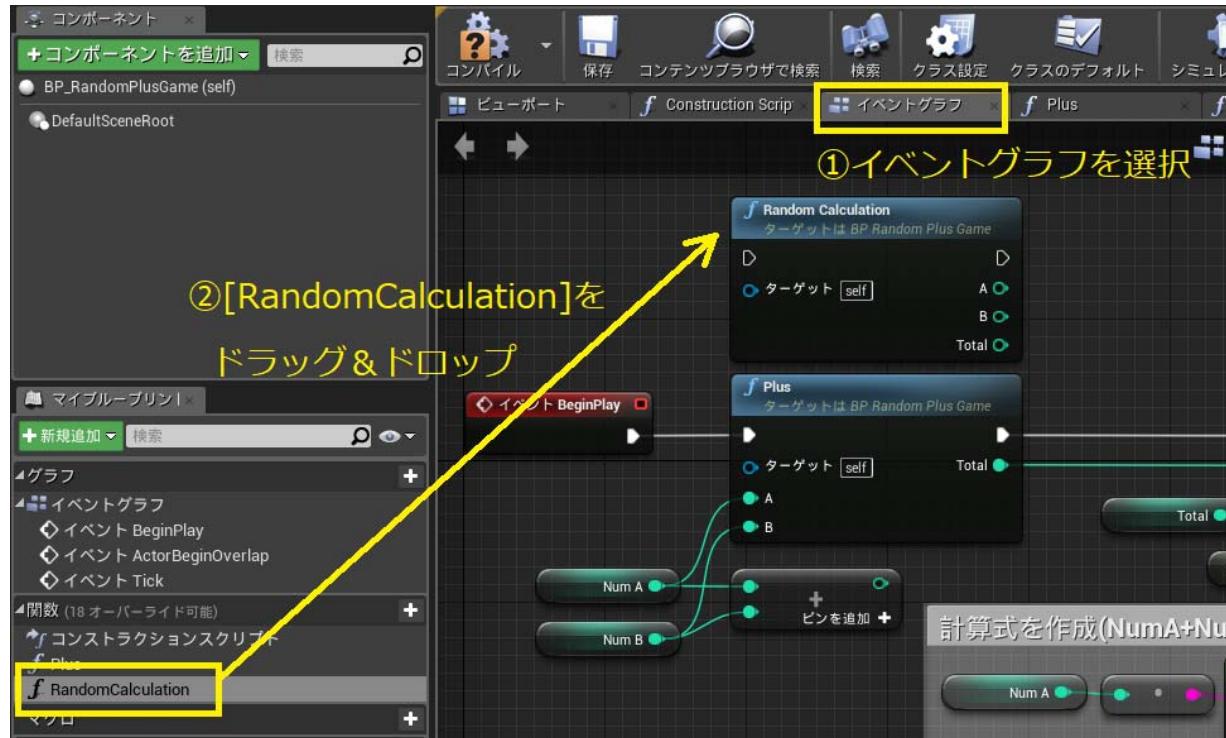
ローカル変数のゲットを追加して足し算を実装する



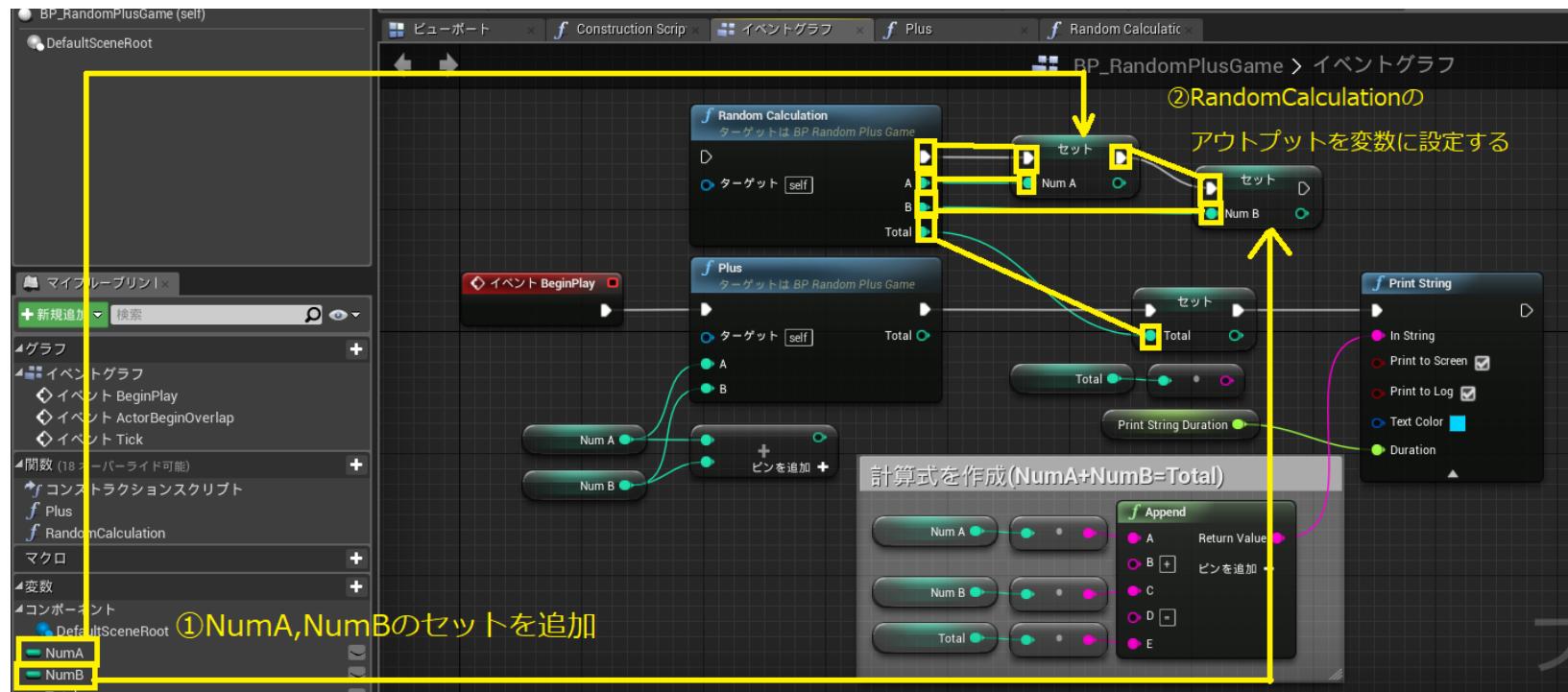
リターンノードに
ランダム値と足し算結果をつなげる



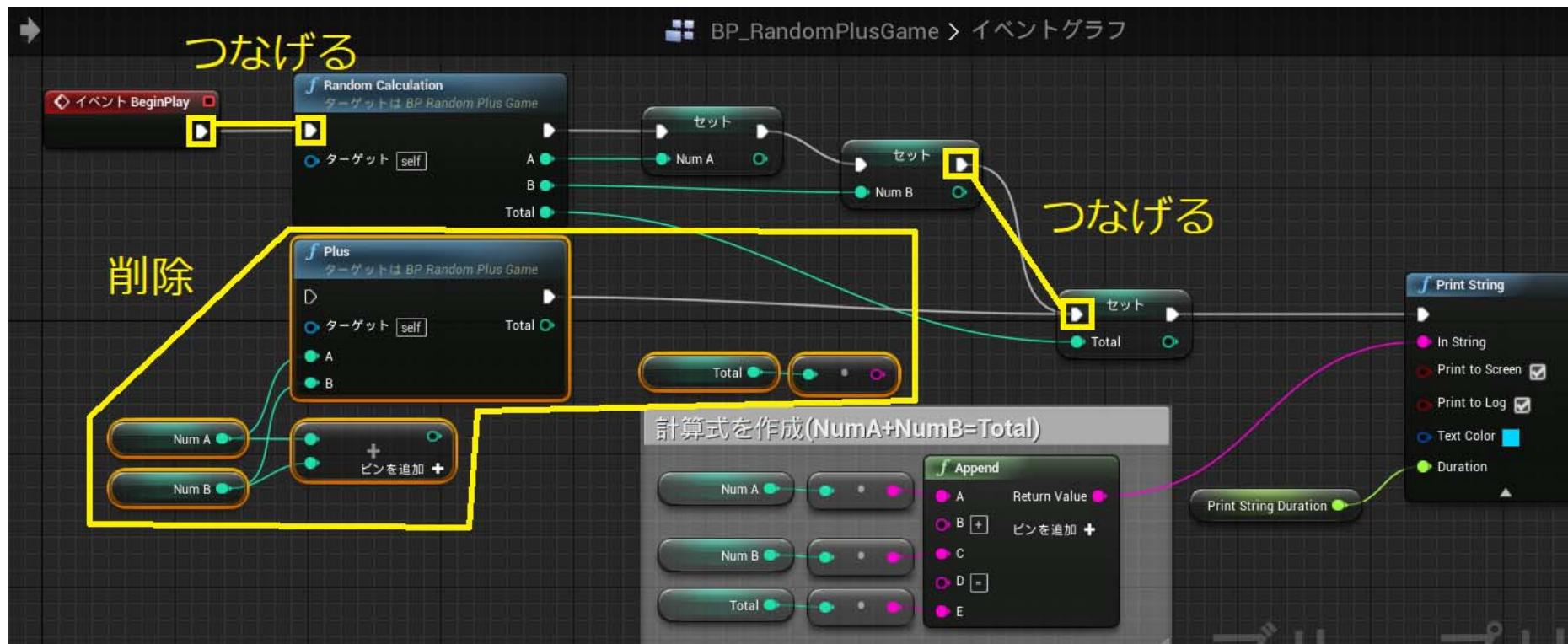
RandomCalculationをイベントグラフに追加



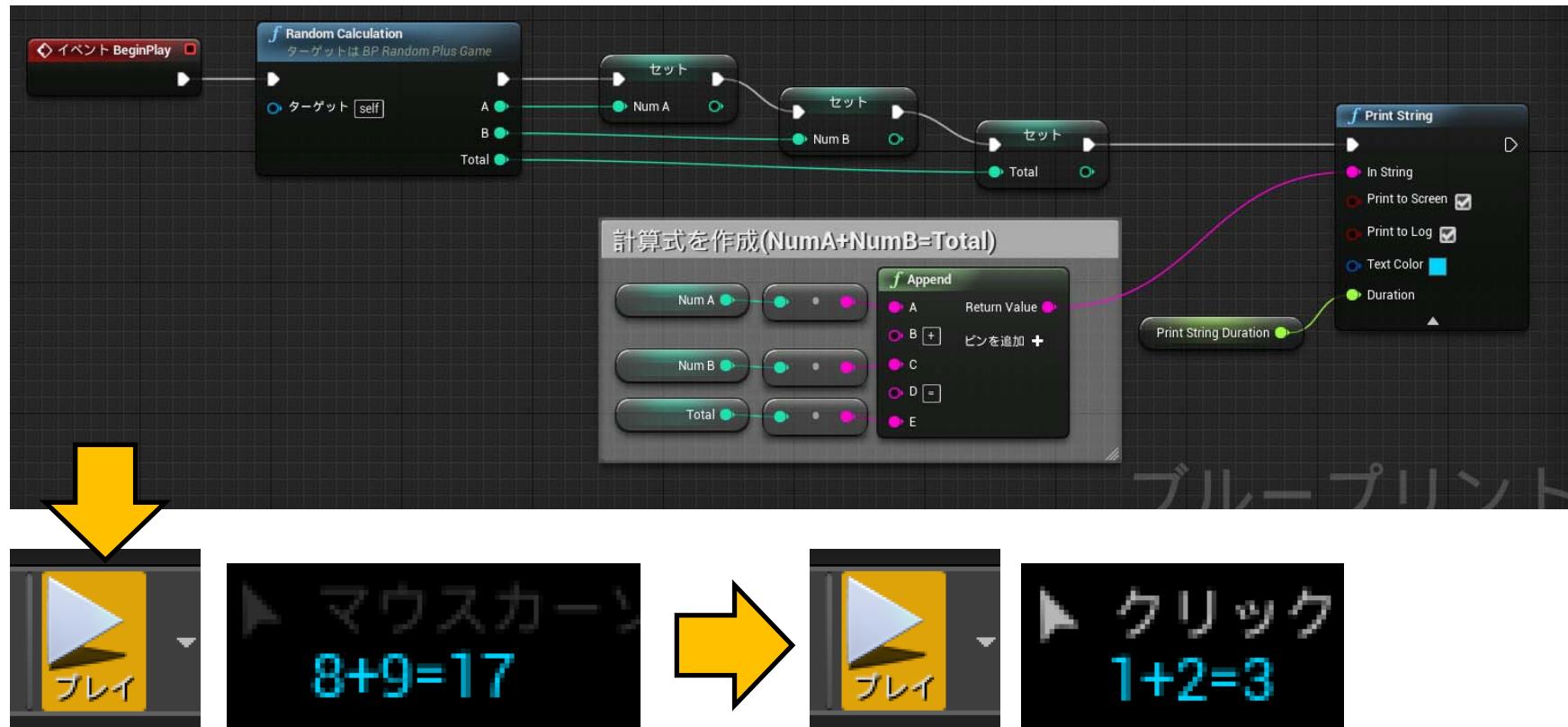
RandomCalculationのアウトプットを 変数に設定する



PlusからRandomCalculationを実行するようにつなぎ変える



プレイするたびに
計算結果が変わることを確認



4. ブループリントで足し算ゲームを作成する

4.1 新規レベル (RandomPlusGame) を作成・保存

4.2 ブループリント(BP_RandomPlusGame)を作成

4.3 足し算の実装

4.4 足し算の引数を変数化

4.5 足し算を関数化

4.6 ランダムの数値を足し算する関数(RandomCalculation)を作成

4.7 キーボード入力で答えを入力

4.8 正解判定

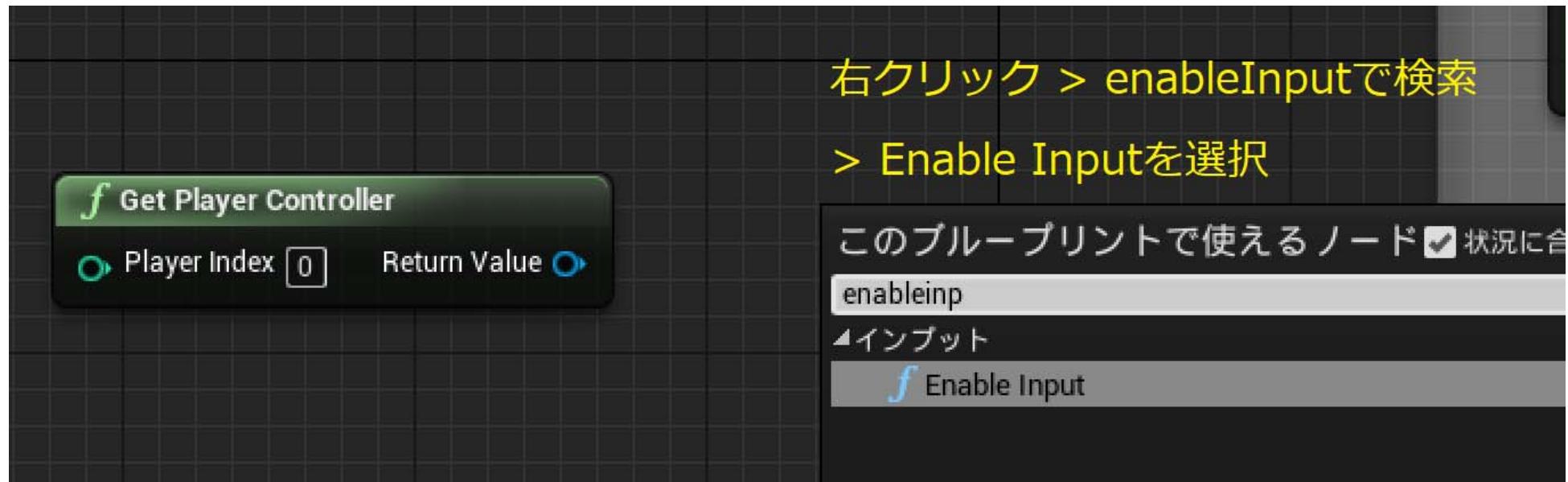
4.9 算数ゲーム(足し算のみ)化

キーボード入力を受け付ける設定 1 Get Player Controllerの追加



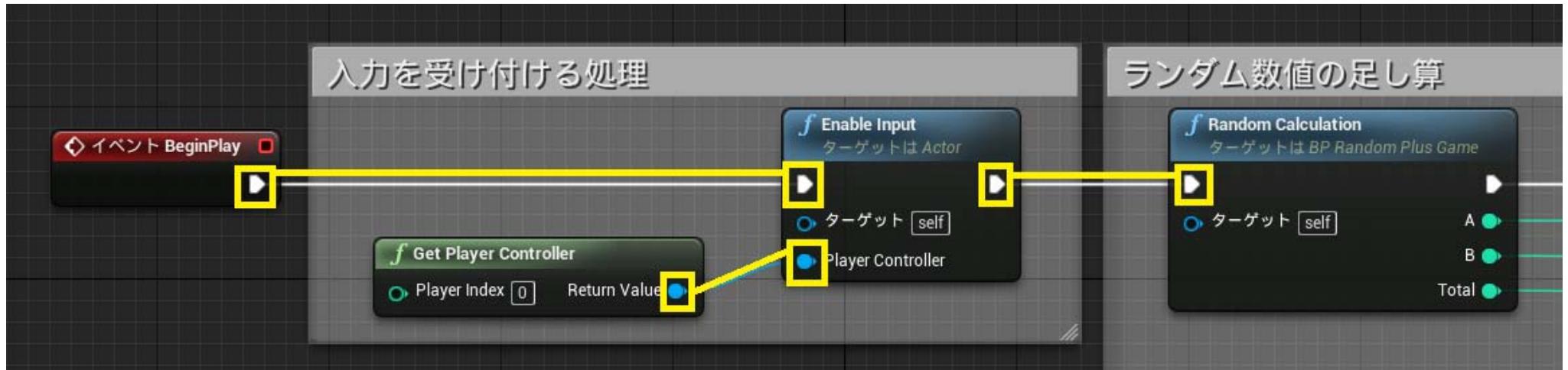
右クリック > gatgameplayerconで検索 > Get Player Controllerを選択

キーボード入力を受け付ける設定 2 Enable Inputを追加する



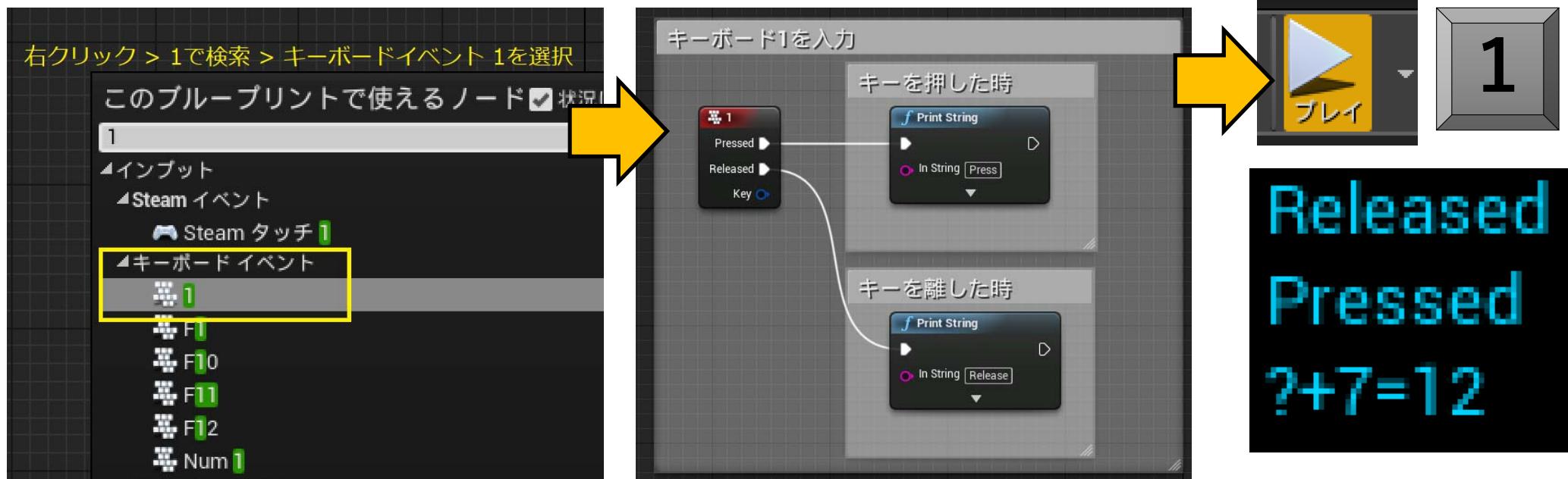
右クリック > enableInputで検索 > Enable Inputを選択

キーボード入力を受け付ける設定 3 入力を受け付ける処理につなげ変える



BeginPlayとRandomCalculationの間にEnable Inputを処理するようにつなげ変える

キーボードイベント 1を追加 キーイベントの確認

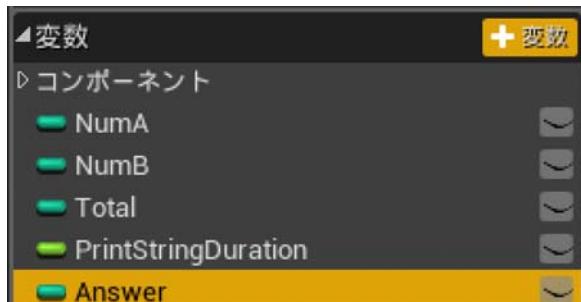


右クリック > 1で検索
>キーボードイベント 1を選択

PrintStringで
Pressed,Releasedの確認を行う

プレイして確認

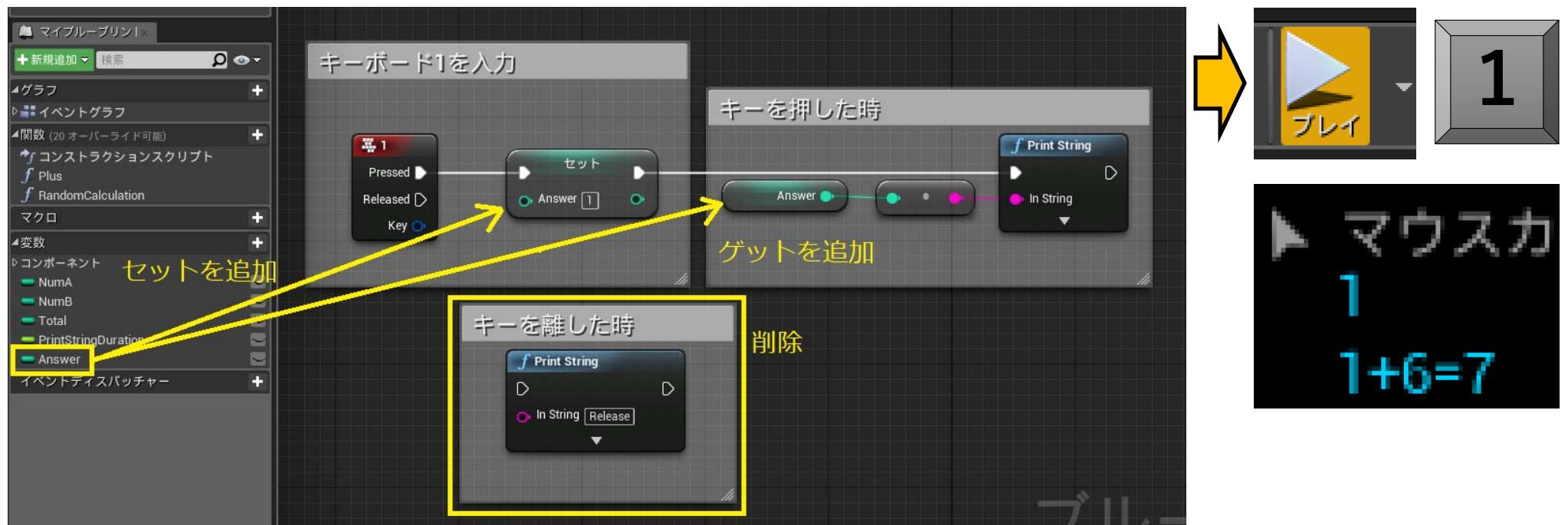
変数: Answerを追加する



[+変数]をクリック
> 名前を[Answer]に設定

变数名	变数の型	デフォルト値
NumA	Integer	8
NumB	Integer	5
Total	Integer	-
PrintStringDuration	Float	10.0
Answer	Integer	-

1を入力した際にAnswerに1をセットする
PrintStringでセットした値を出力



Answerのセットを追加し、1イベントのPressed時にAnswerに1をセットする
Answerのゲットを追加し、PrintStringでAnswerの値を出力する

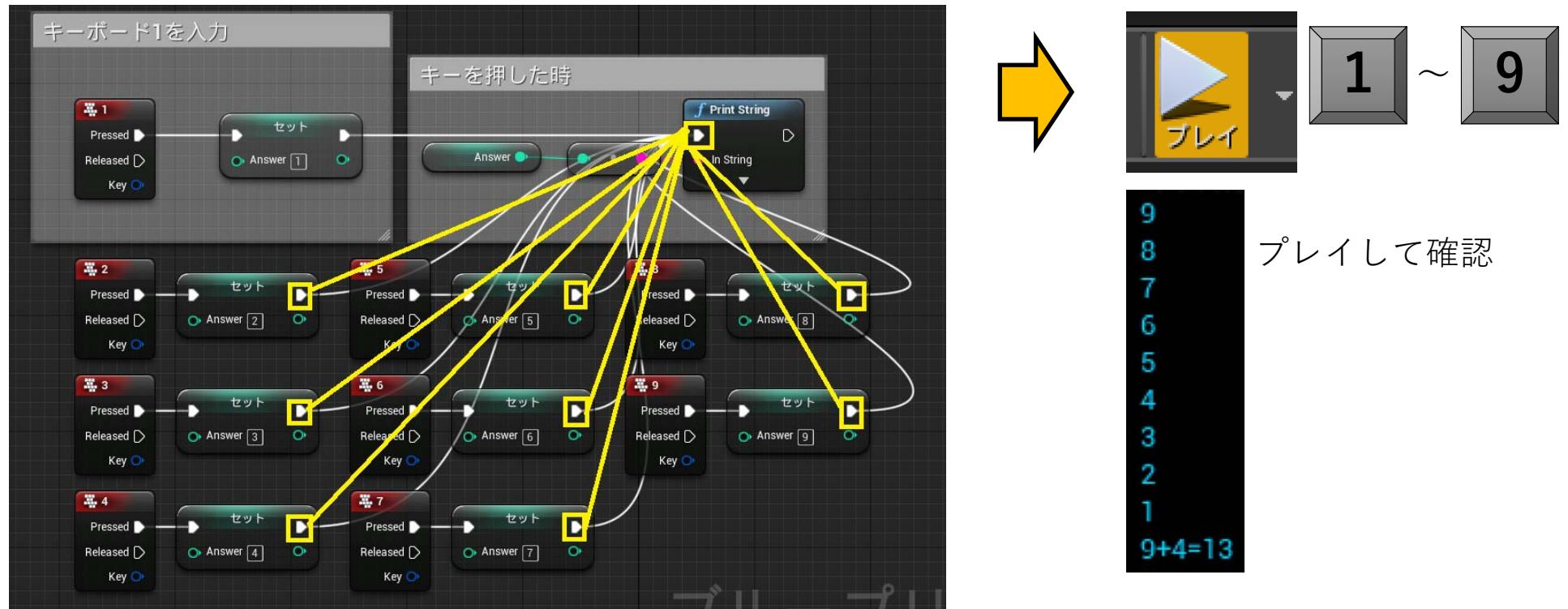
プレイして確認

キーボード2~9のイベントを作成 Answerに値を設定する



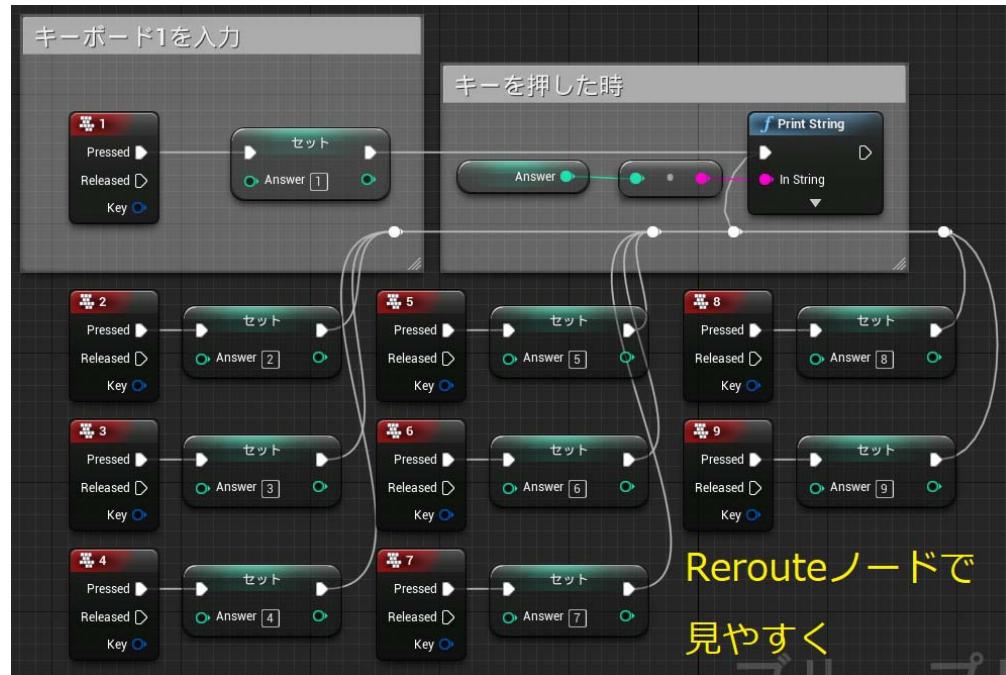
キーボード1イベントと同様に、キーボード2~9 イベントを追加して、Answerに値をセットする

イベント2~9のAnswerセット後に
PrintStringを呼び出すようにつなげる



キーボード2~9 イベントのAnswerにセットした後にPrintStringにつなげる

Rerouteノードで処理を見やすくする



Rerouteノードを使うと線の中継地点をつくれる
ノードに被っている線を被らなくしたり、
同じ処理をする線をまとまることができる



[Rerouteノードの追加方法]

- Routeノードを追加したい箇所をダブルクリック
- 右クリック > Rerouteノードを追加

4. ブループリントで足し算ゲームを作成する

4.1 新規レベル (RandomPlusGame) を作成・保存

4.2 ブループリント(BP_RandomPlusGame)を作成

4.3 足し算の実装

4.4 足し算の引数を変数化

4.5 足し算を関数化

4.6 ランダムの数値を足し算する関数(RandomCalculation)を作成

4.7 キーボード入力で答えを入力

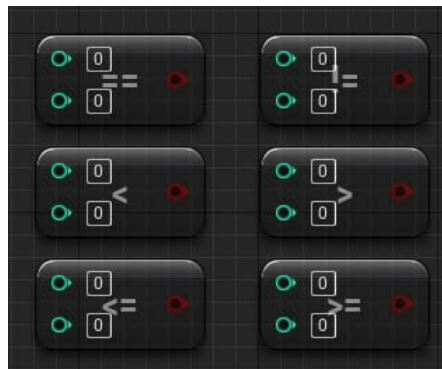
4.8 正解判定

4.9 算数ゲーム(足し算のみ)化

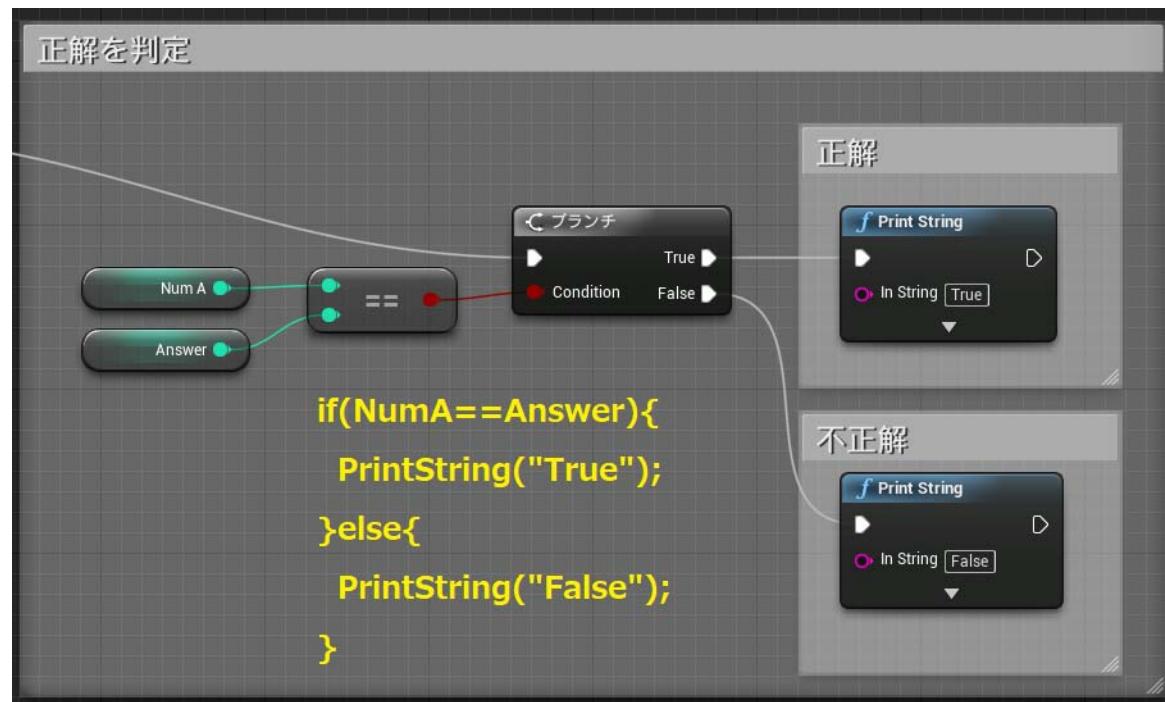
ブループリントのif文



if = ブランチ

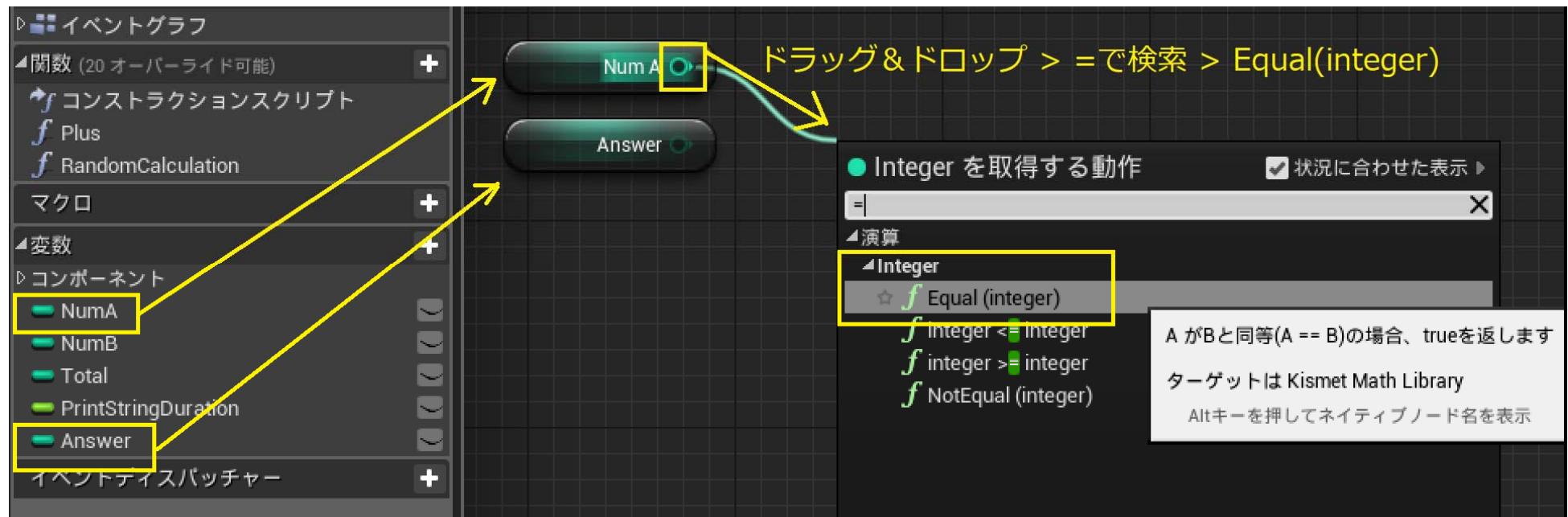


比較演算子ノード



ブループリントのIF文の組み方
比較演算子ノードで比較した結果を
ブランチのConditionに繋げる

NumA,Answerが一致するか比較する 1
NumA,AnswerのゲットとEqual(integer)を追加する



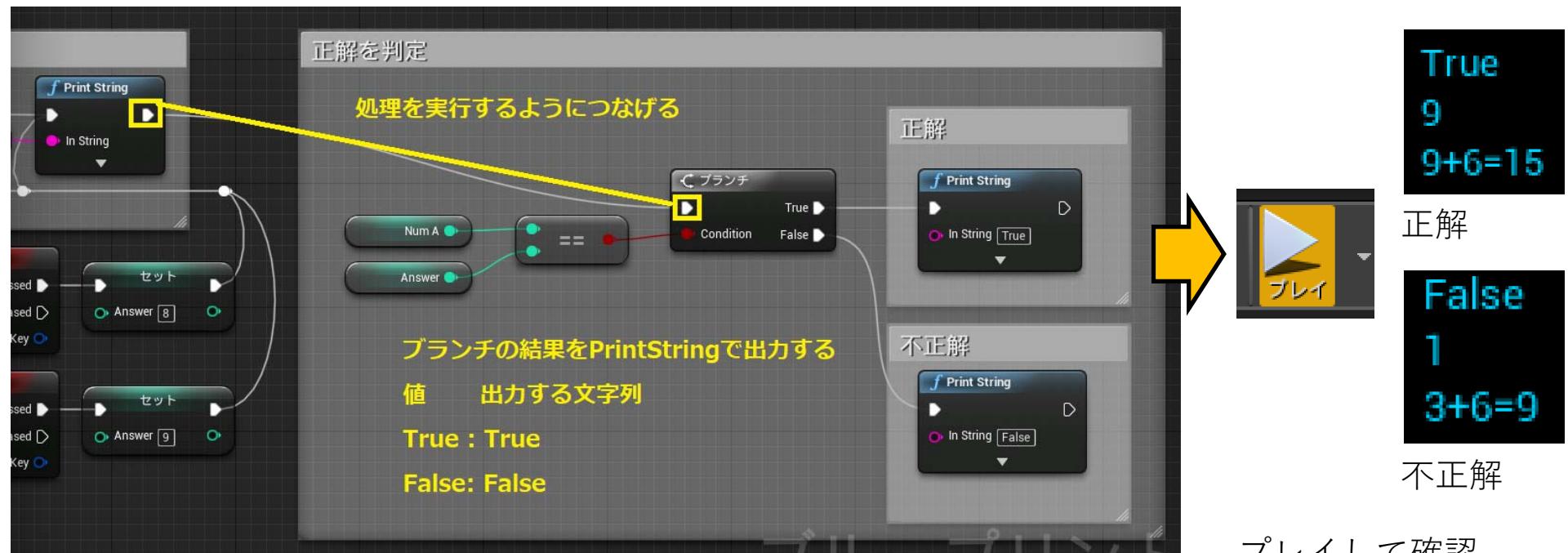
1. NumA,Answerのゲットを追加する
2. Equal(integer)を追加する

NumA,Answerが一致するか比較する
2
ブランチを追加する



右クリック > branch or ブランチで検索 > ブランチを選択

NumA,Answerが一致するか比較する2
ブランチの結果をPrintStringで出力する



1. Equal(integer)の結果をブランチのConditionにつなげる
2. ブランチの結果をPrintStringで出力する
3. ブランチの処理を実行するようにキーイベントのPrintStringとつなげる

プレイして確認
入力したキーに
応じて出力結果が変わる

4. ブループリントで足し算ゲームを作成する

4.1 新規レベル (RandomPlusGame) を作成・保存

4.2 ブループリント(BP_RandomPlusGame)を作成

4.3 足し算の実装

4.4 足し算の引数を変数化

4.5 足し算を関数化

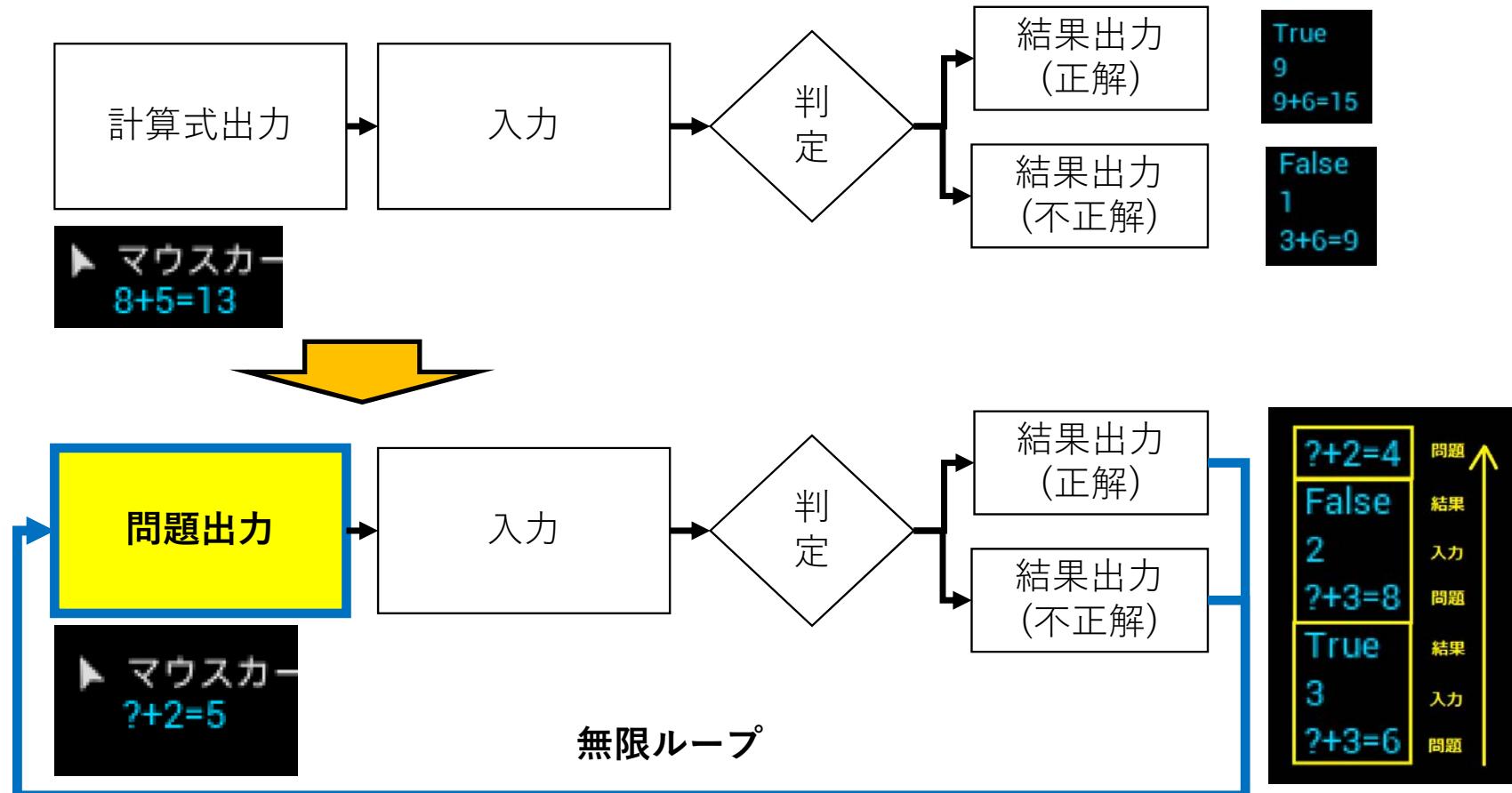
4.6 ランダムの数値を足し算する関数(RandomCalculation)を作成

4.7 キーボード入力で答えを入力

4.8 正解判定

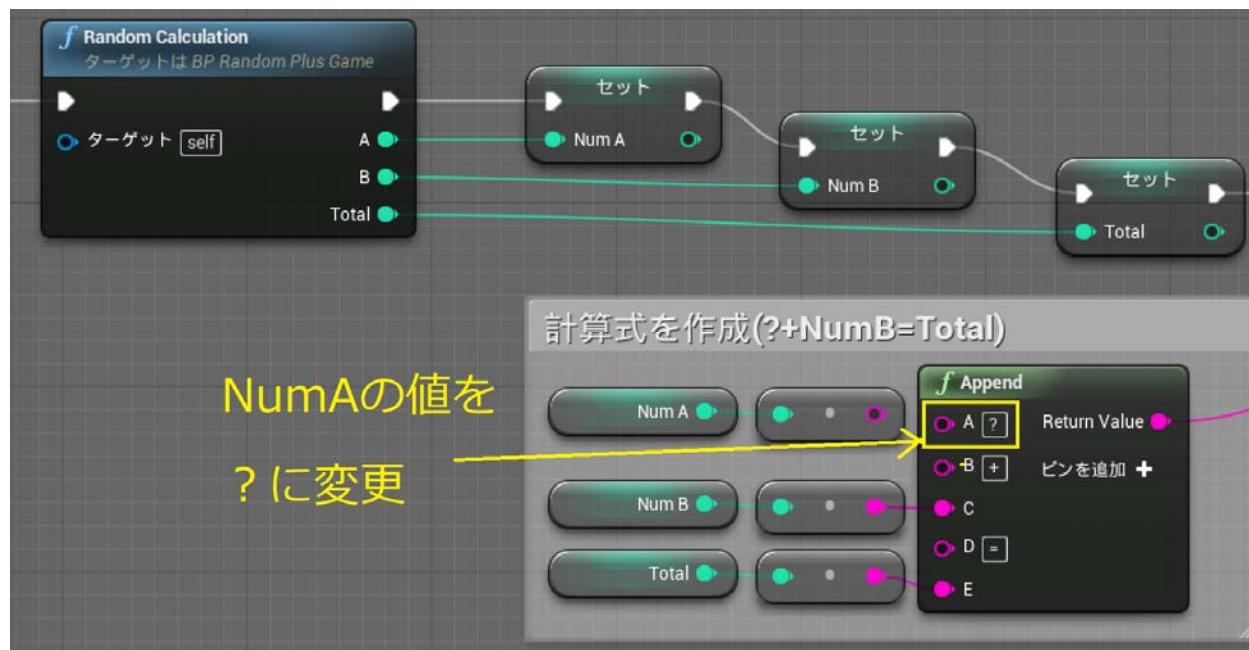
4.9 算数ゲーム(足し算のみ)化

ゲーム化（計算式の問題化、無限ループ）

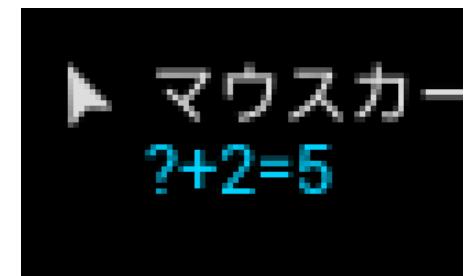
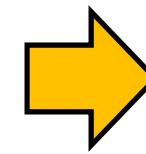


計算式の問題化

計算式のNumAの値を?に変更する



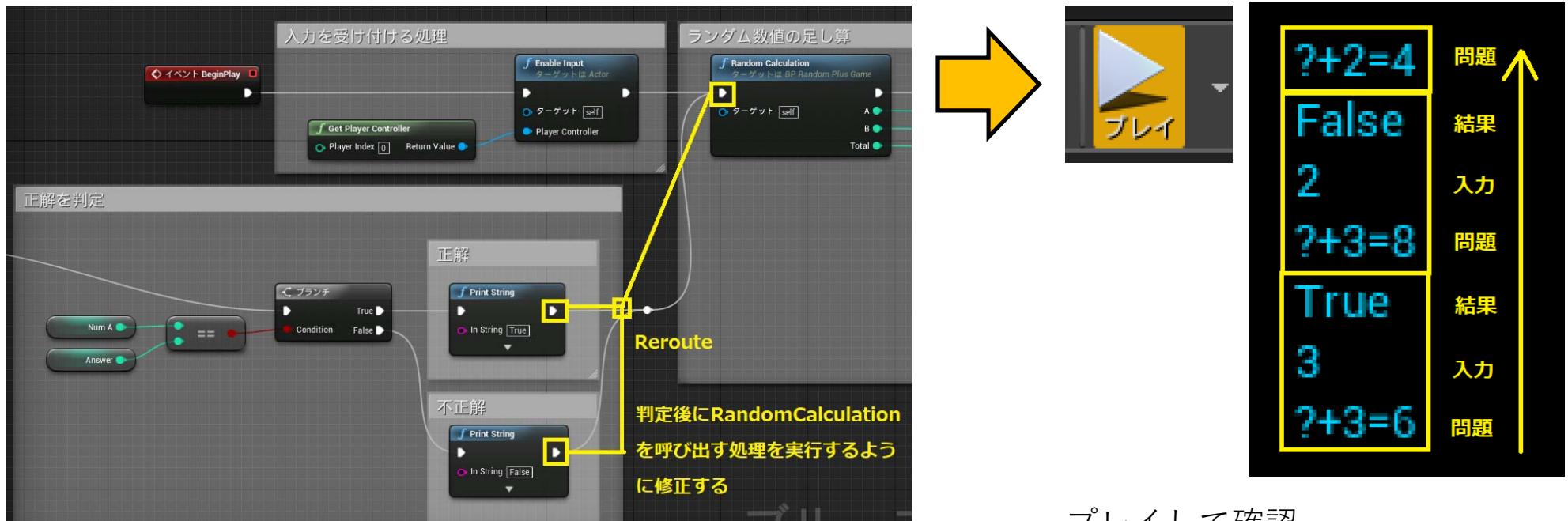
計算式のNumAの文字列を追加しているAppendのAの値を[?]に変更する



プレイして確認
 $?+(NumB)=(Total)$

無限ループ

正解を判定後RandomCalculationを呼び出す処理を実行する



正解、不正解を出力後にRandomCalculationを呼び出す

プレイして確認
問題 > 入力 > 結果を繰り返す

5. 計算方法(足し算、引き算、掛け算、割り算)をランダムに修正する

5. 計算方法(足し算、引き算、掛け算、割り算)をランダムに修正する

5.1 新規レベル (RandomCalcGame) を作成・保存

5.2 BP_RandomPlusGameを複製して、BP_RandomCalcGameを作成

5.3 引き算、掛け算、割り算を実装

5.4 関数RandomCalculationの修正

5.5 算数ゲームの計算方法のランダム対応

5. 計算方法(足し算、引き算、掛け算、割り算)をランダムに修正する

5.1 新規レベル (RandomCalcGame) を作成・保存

5.2 BP_RandomPlusGameを複製して、BP_RandomCalcGameを作成

5.3 引き算、掛け算、割り算を実装

5.4 関数RandomCalculationの修正

5.5 算数ゲームの計算方法のランダム対応

新規レベルの作成



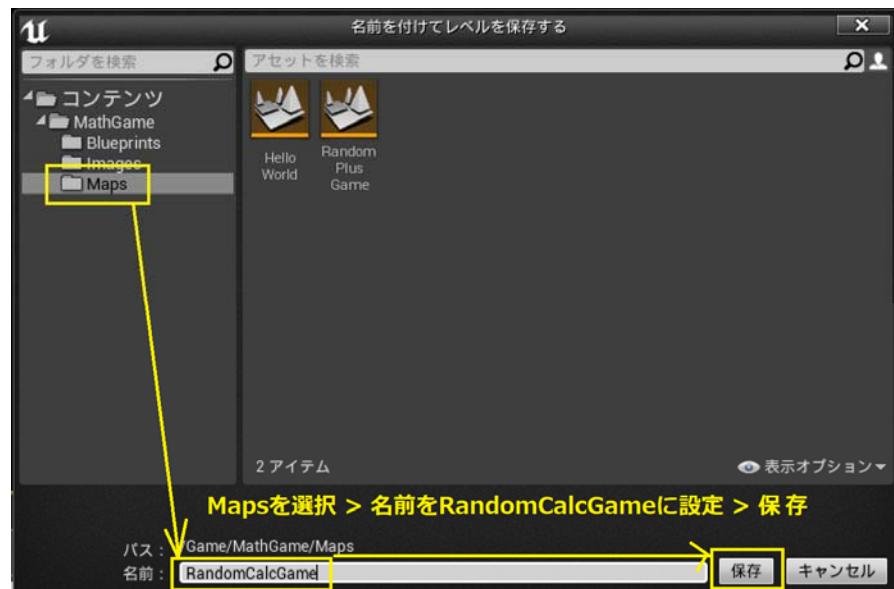
ファイル > 新規レベル
ショートカット : Ctrl+N

[空のレベル]を選択する

現在のレベルを保存
名前を[RandomCalcGame]に設定して保存



ファイル > 現在のレベルを保存
ショートカット : Ctrl+S



Maps フォルダを選択
> 名前を[RandomCalcGame]に設定
> [保存]をクリック

5. 計算方法(足し算、引き算、掛け算、割り算)をランダムに修正する

5.1 新規レベル (RandomCalcGame) を作成・保存

5.2 BP_RandomPlusGameを複製して、BP_RandomCalcGameを作成

5.3 引き算、掛け算、割り算を実装

5.4 関数RandomCalculationの修正

5.5 算数ゲームの計算方法のランダム対応

BP_RandomPlusGameを複製し、 BP_RandomCalcGameを作成

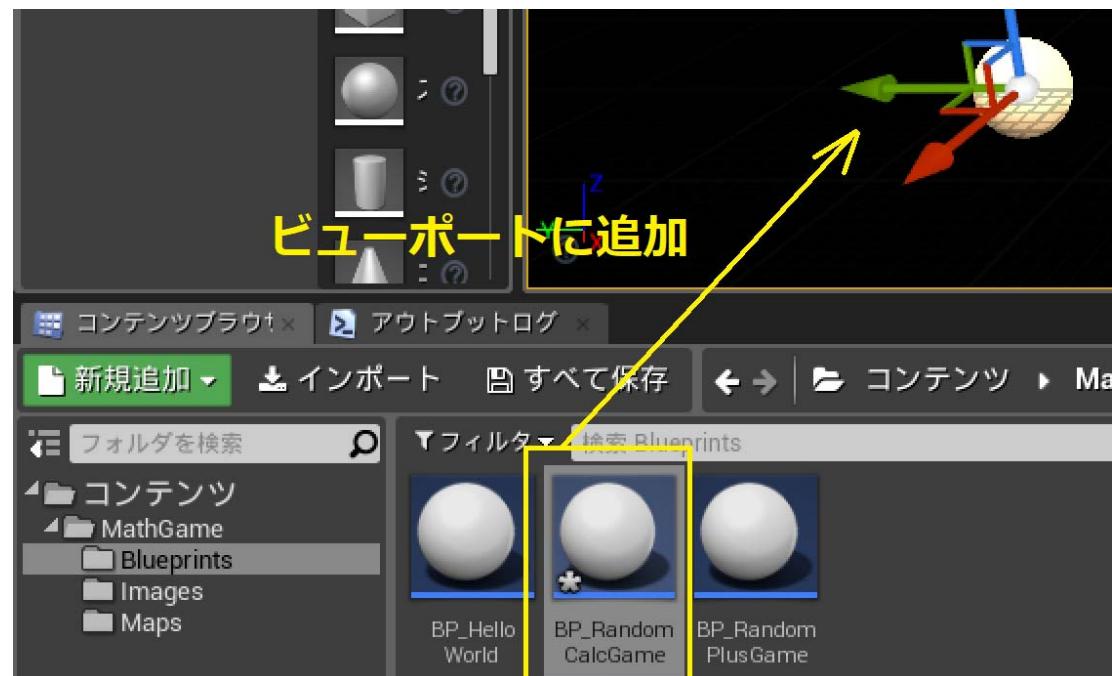


BP_RandomPlusGameを右クリック > 複製



名前を[BP_RandomCalcGame]に設定

ビューポートに追加し、BP_RandomPlusGameと
同様の処理をすることを確認



BP_RandomCalcGameをビューポートにドラッグ&ドロップ



プレイして確認
BP_RandomPlusGameと
同様の処理

5. 計算方法(足し算、引き算、掛け算、割り算)をランダムに修正する

5.1 新規レベル (RandomCalcGame) を作成・保存

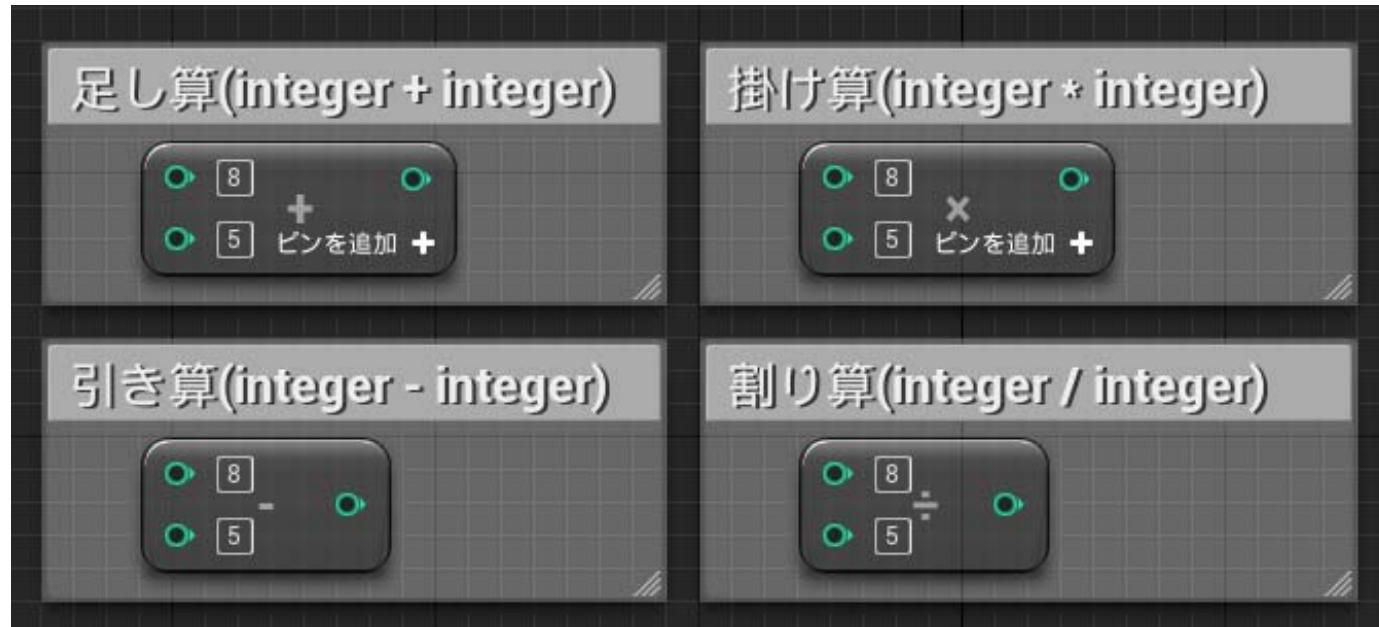
5.2 BP_RandomPlusGameを複製して、BP_RandomCalcGameを作成

5.3 引き算、掛け算、割り算を実装

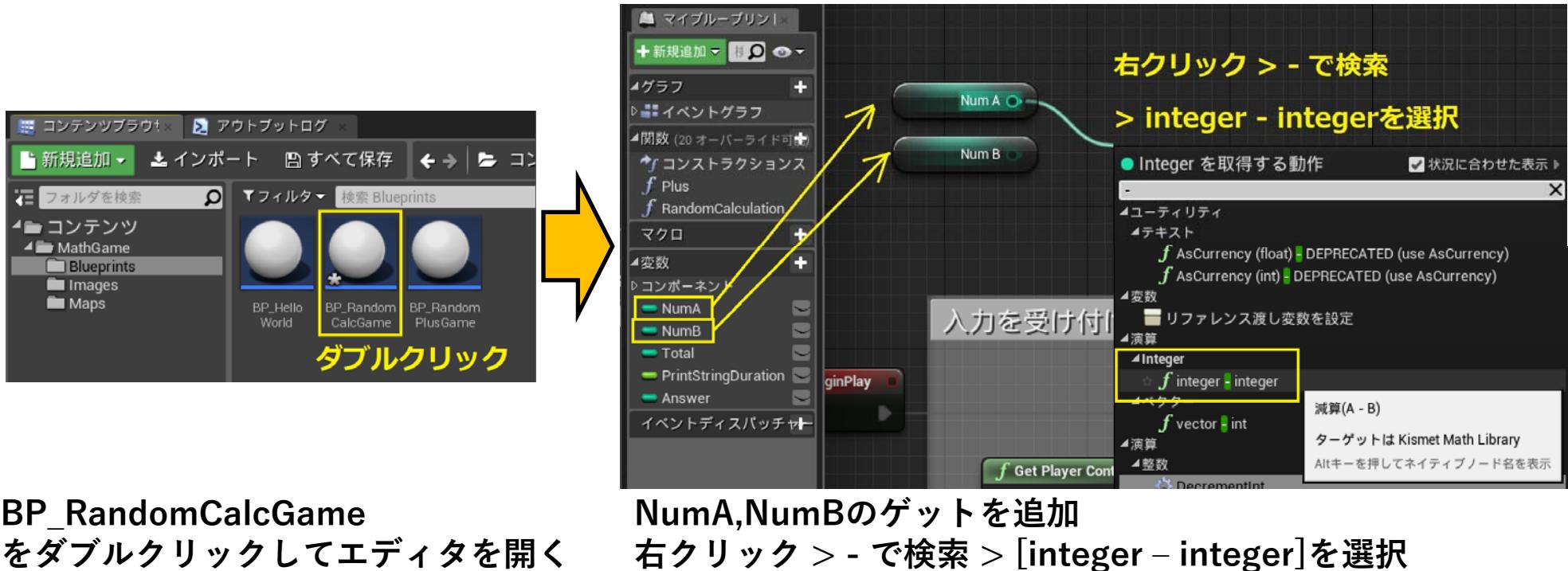
5.4 関数RandomCalculationの修正

5.5 算数ゲームの計算方法のランダム対応

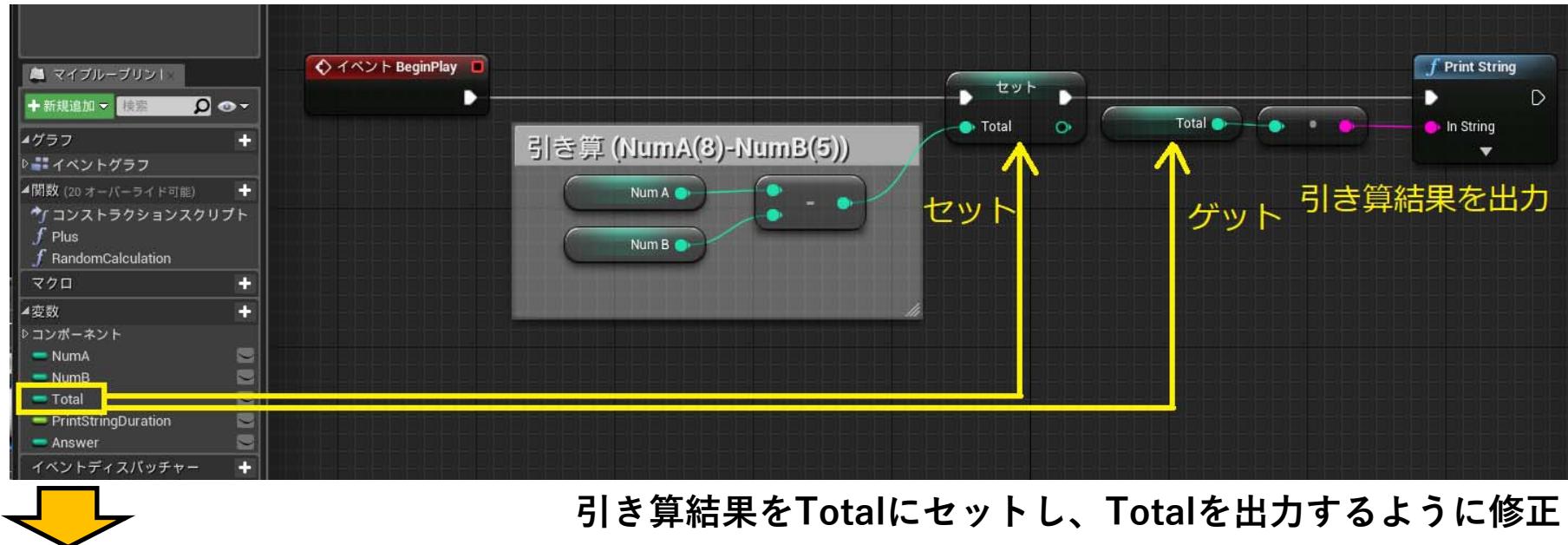
計算ノード



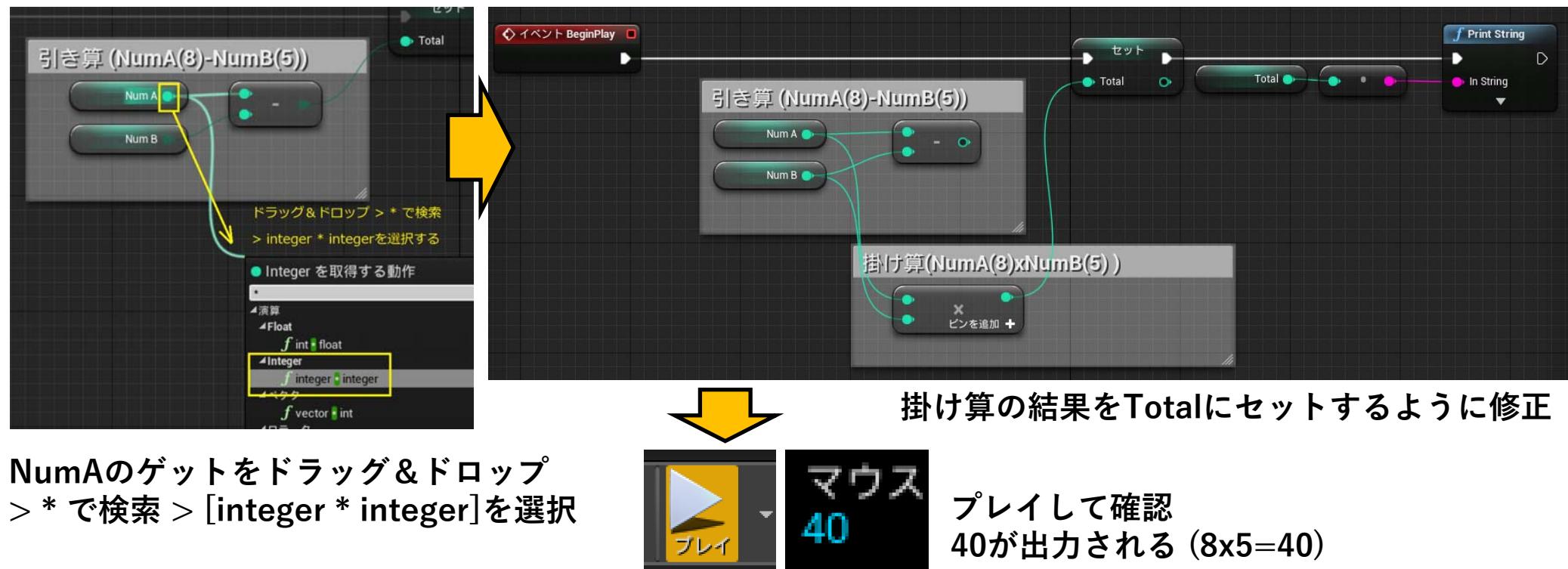
BP_RandomCalcGameを開く
NumA,NumBのゲットを追加
引き算するノード(integer - integer)を追加する



引き算の結果を出力するように修正

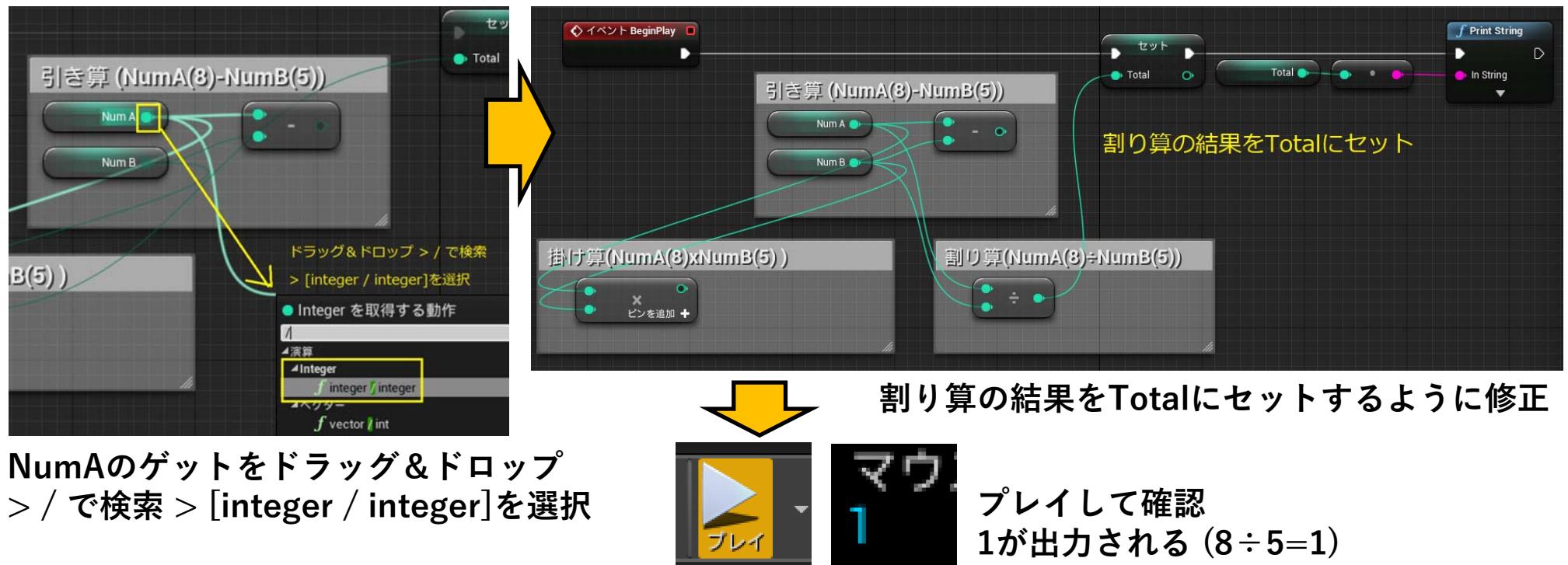


掛け算ノードの結果を出力



NumAのゲットをドラッグ&ドロップ
> * で検索 > [integer * integer]を選択

割り算ノードの結果を出力



Integerは整数しか保持できない
変数：Totalの型をFloatに変更する



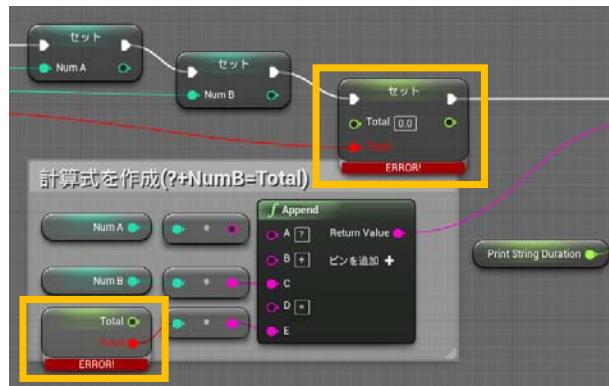
Totalの変数の型を
Floatに変更する

変数名	変数の型	デフォルト値
NumA	Integer	8
NumB	Integer	5
Total	Float	-
PrintStringDuration	Float	10.0
Answer	Integer	-

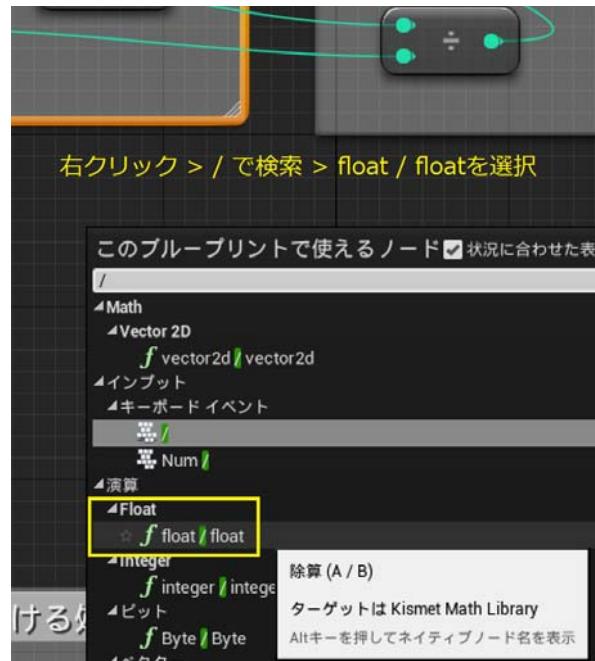


integer / integer : $8 \div 5 = 1$
float / float : $8 \div 5 = 1.6$

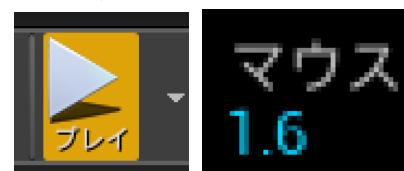
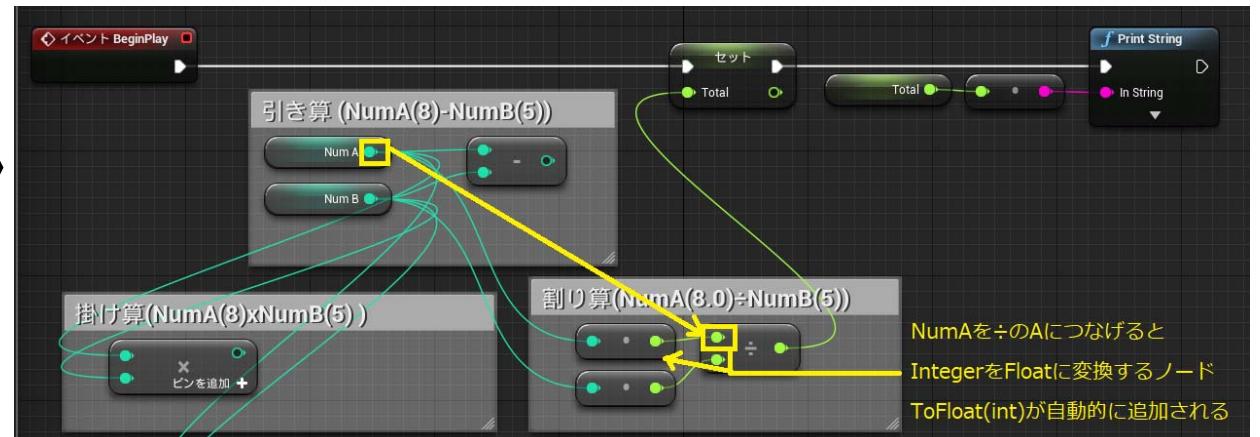
Total をIntegerとしてにつないでいる箇所
がError(赤く)なるので接続を切る



割り算(float / float)の結果を出力



右クリック > / で検索
> float / floatを選択



割り算(float / float)の結果を
Totalにセットするように修正

プレイして確認
1.6が出力される ($8 \div 5 = 1.6$)

変数の型によって入れられる値が違う

変数の型	概要	具体的な値
Boolean	True(真)かFalse(偽)の二つの値が入る	True, False
Byte	正の整数の数値が入る	0~255
Integer	整数の数値が入る	-1, 0, 1, 2, 3, 4, 5 ··· 100
Float	少数点の値が入る	0.0553, 101.2887, -78.322
Name	文字列（ネットワークで使用）が入る	半角英数(0-9,A-Z,a-z), 日本語
String	文字列（プログラムで使用）が入る	半角英数(0-9,A-Z,a-z), 日本語
Text	文字列（ウィジェット, ローカリゼーションで使用）が入る	半角英数(0-9,A-Z,a-z), 日本語
Vector	FloatがXYZ(座標、スケール、色RGBで使用)に入る	位置(XYZ)、 色(RGB)
Rotator	回転情報のFloatがXYZ(X:Roll, Y:Pitch, Z:Yaw)に入る	回転情報(XYZ)
Transform	位置(Vector)、回転(Rotator)、スケール(Vector)が入る	位置, 回転, スケール

5. 計算方法(足し算、引き算、掛け算、割り算)をランダムに修正する

5.1 新規レベル (RandomCalcGame) を作成・保存

5.2 BP_RandomPlusGameを複製して、BP_RandomCalcGameを作成

5.3 引き算、掛け算、割り算を実装

5.4 関数RandomCalculationの修正

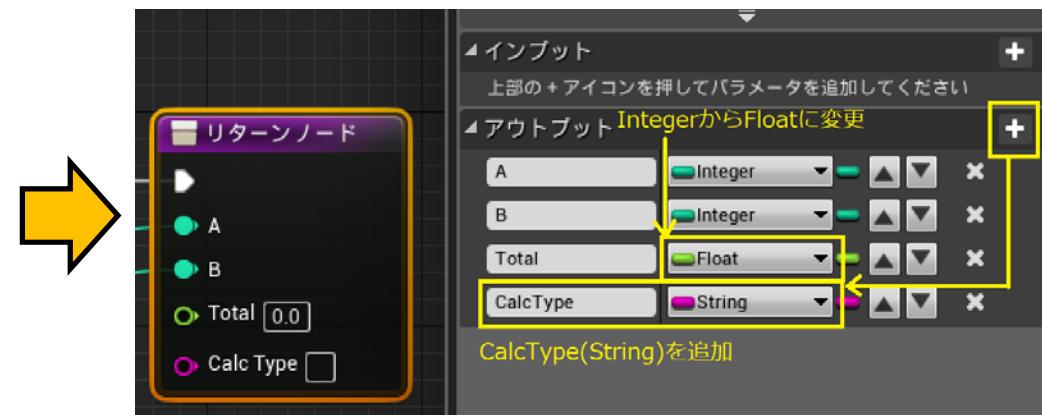
5.5 算数ゲームの計算方法のランダム対応

関数:RandomCalculationの修正 アウトプットにCalcType(String)を追加

関数名	Input/Output	パラメータ名	型
RandomCalculation	Output	A	Integer
	Output	B	Integer
	Output	Total	Float
	Output	CalcType	String



[RandomCalculation]をダブルクリック



アウトプットのパラメータを設定する

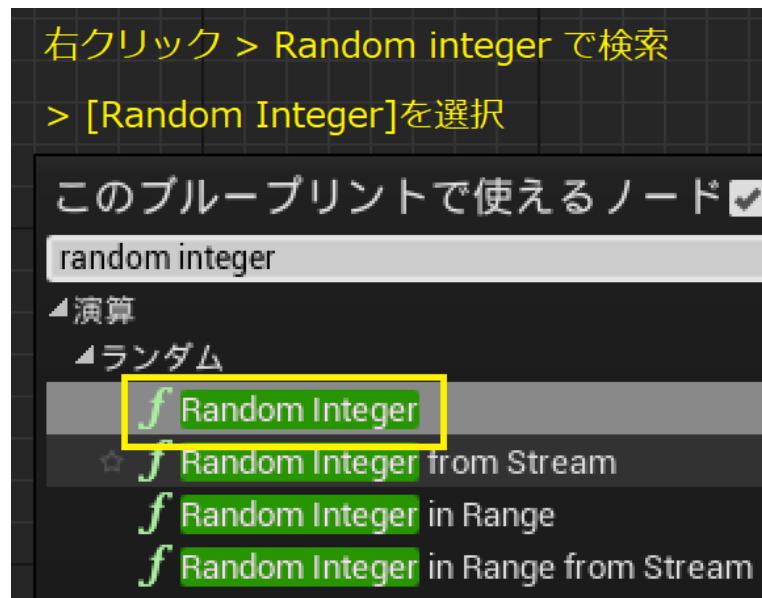
ローカル変数を追加する



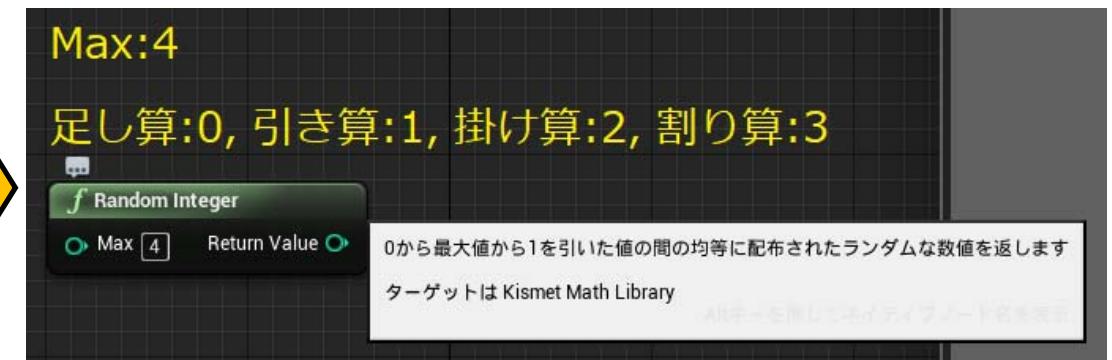
ローカル変数名	変数の型
LocalNumA	Integer
LocalNumB	Integer
LocalTotal	Float
LocalCalcType	String

ローカル変数を追加する
(ローカル変数は関数内のみ有効)

計算方法をランダムで決める



右クリック > random integer で検索
> Random Integerを選択



Maxに 4 を設定

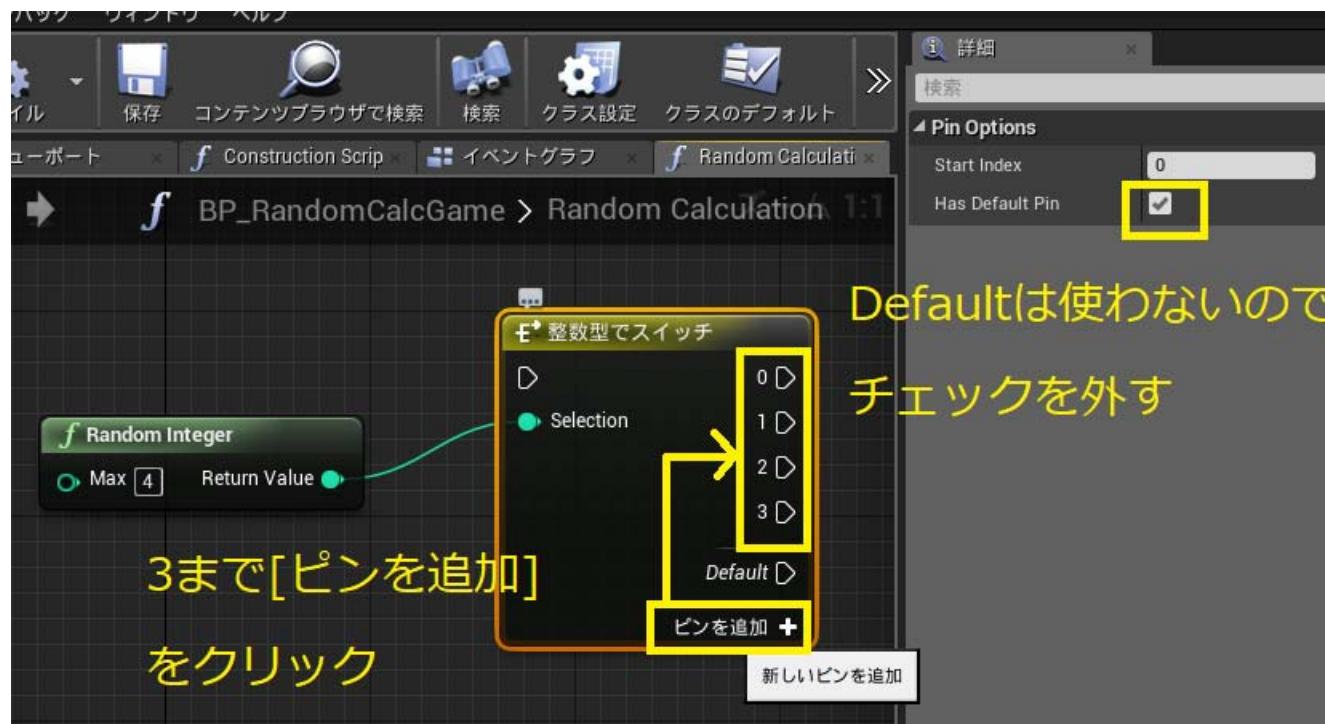
0:足し算
1:引き算
2:掛け算
3:割り算

整数型のスイッチを追加

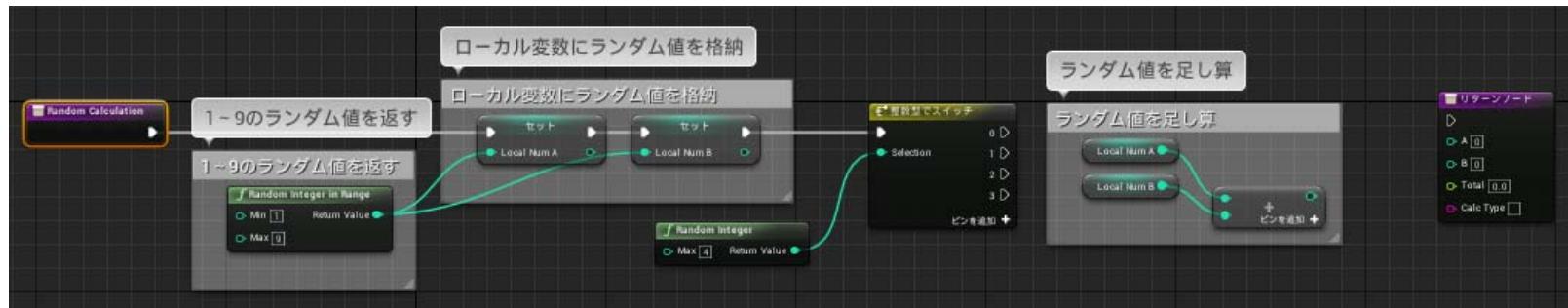


Random IntegerのReturn Valueを ドラッグ&ドロップ> switch で検索
> [整数型でスイッチ]を選択

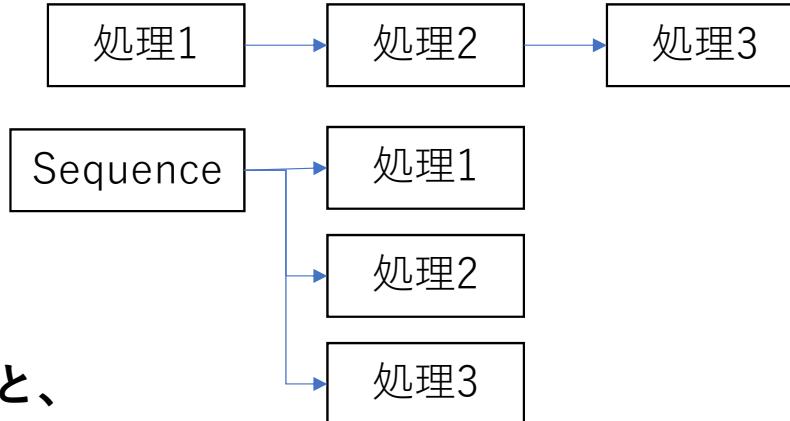
スイッチのピンを追加する
Defaultは使わないのでチェックを外す



Sequenceノードを使って横への間延びを 縦に順々に処理をするように修正

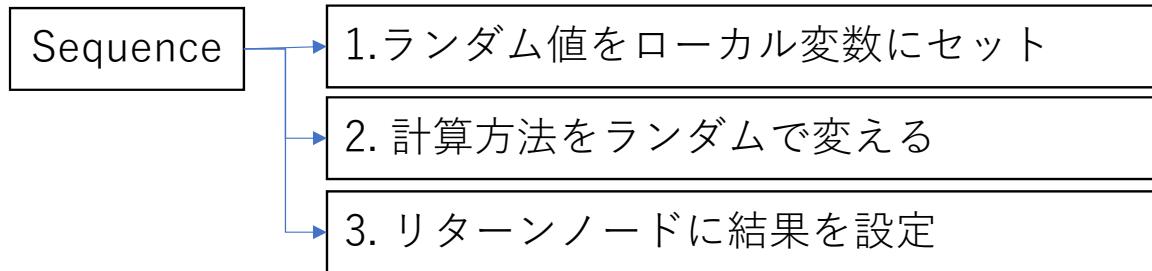
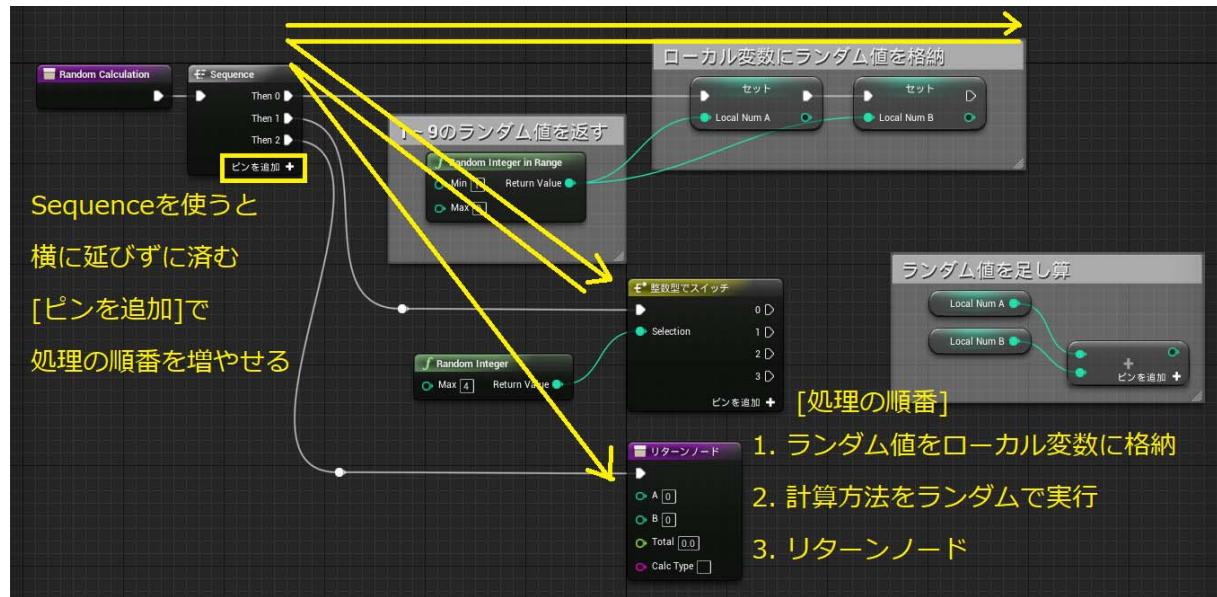
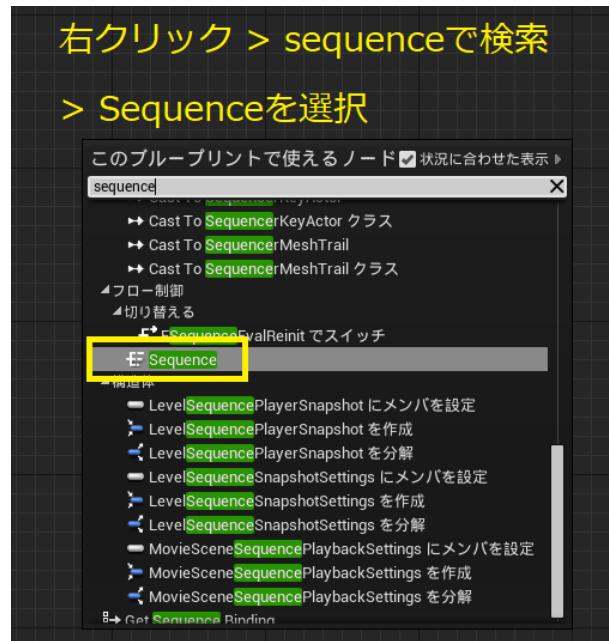


ブループリントを組んでいくと横に間延びしていく



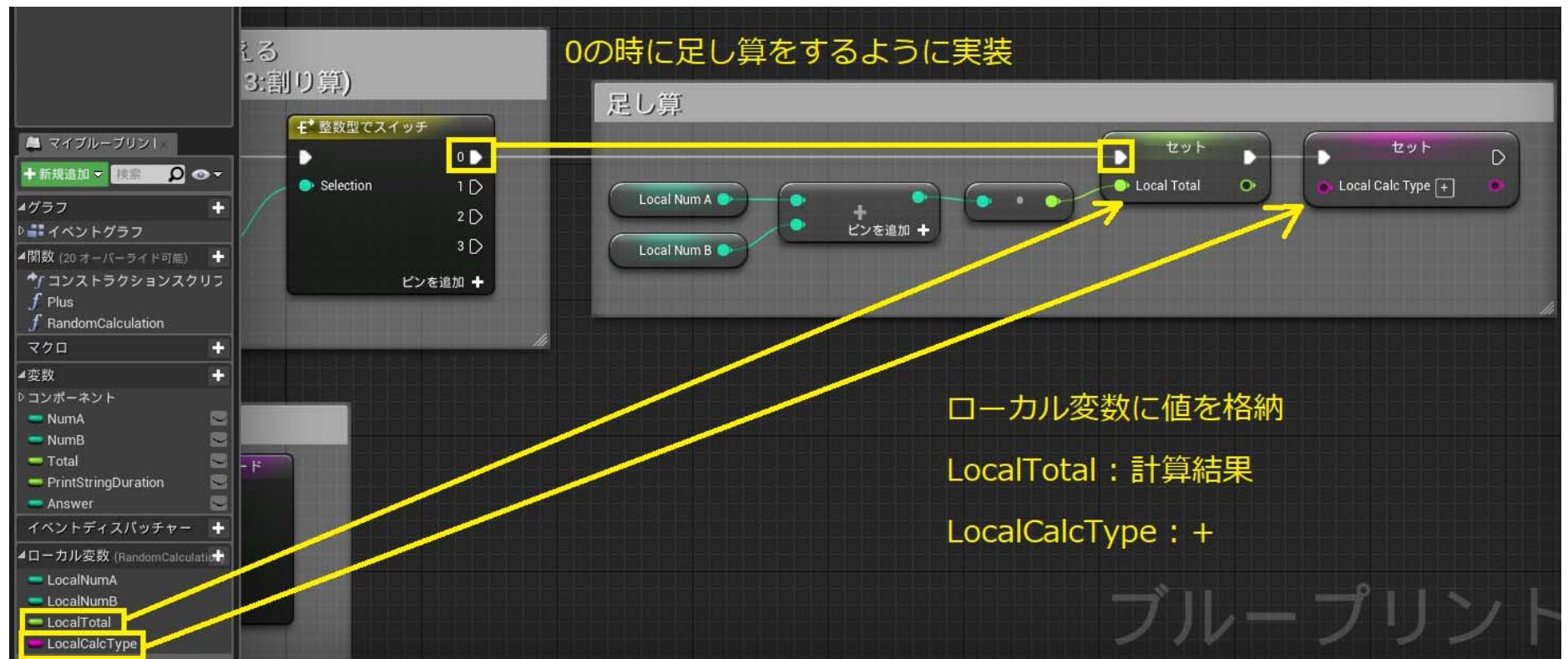
Sequenceノードを使うと、
処理を縦にかける

Sequenceノードを追加して処理を3段階に分ける

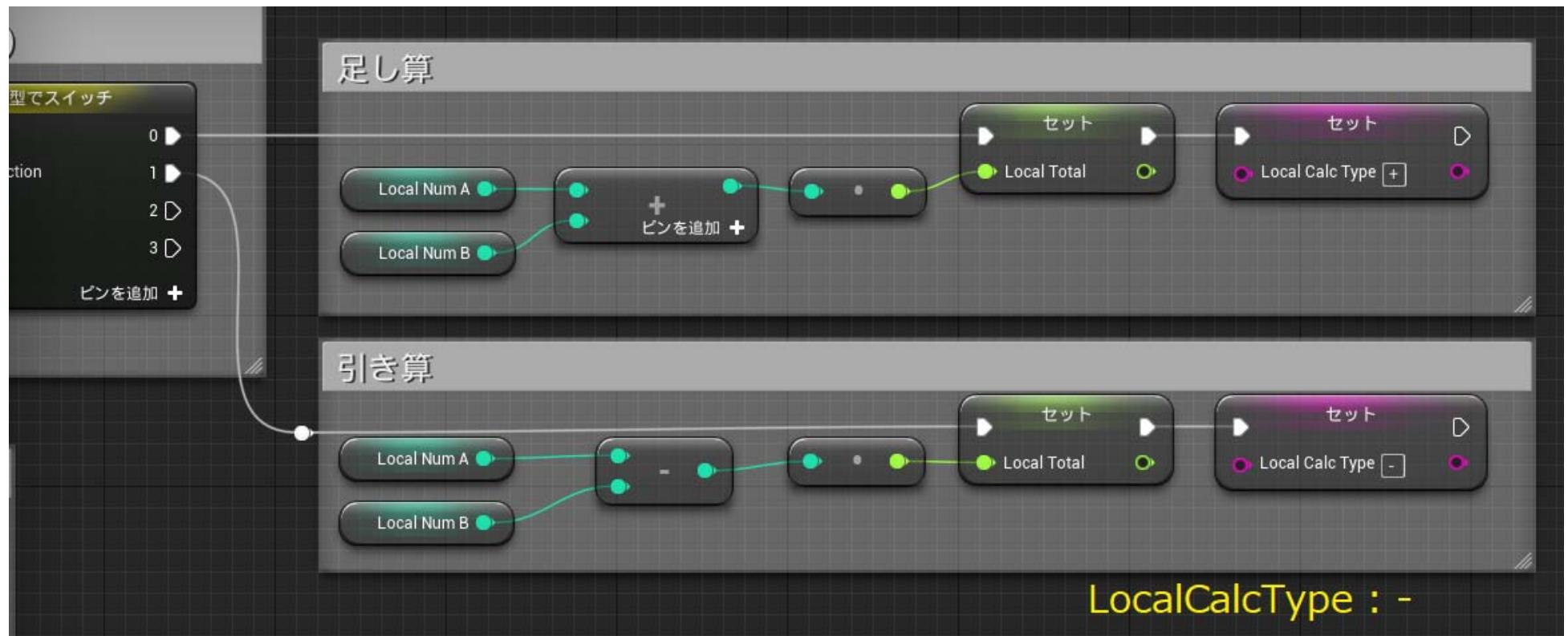


右クリック > sequence で検索
> Sequenceを選択

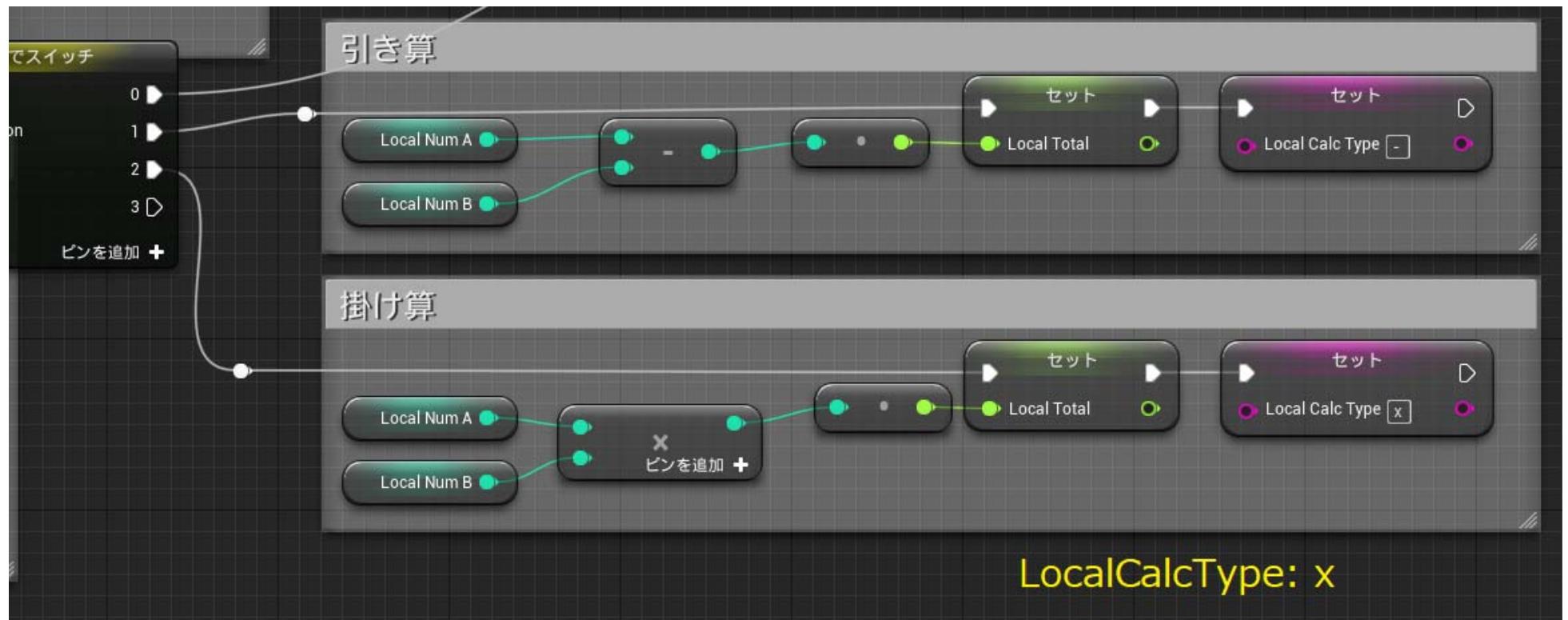
0の時に足し算をするように実装



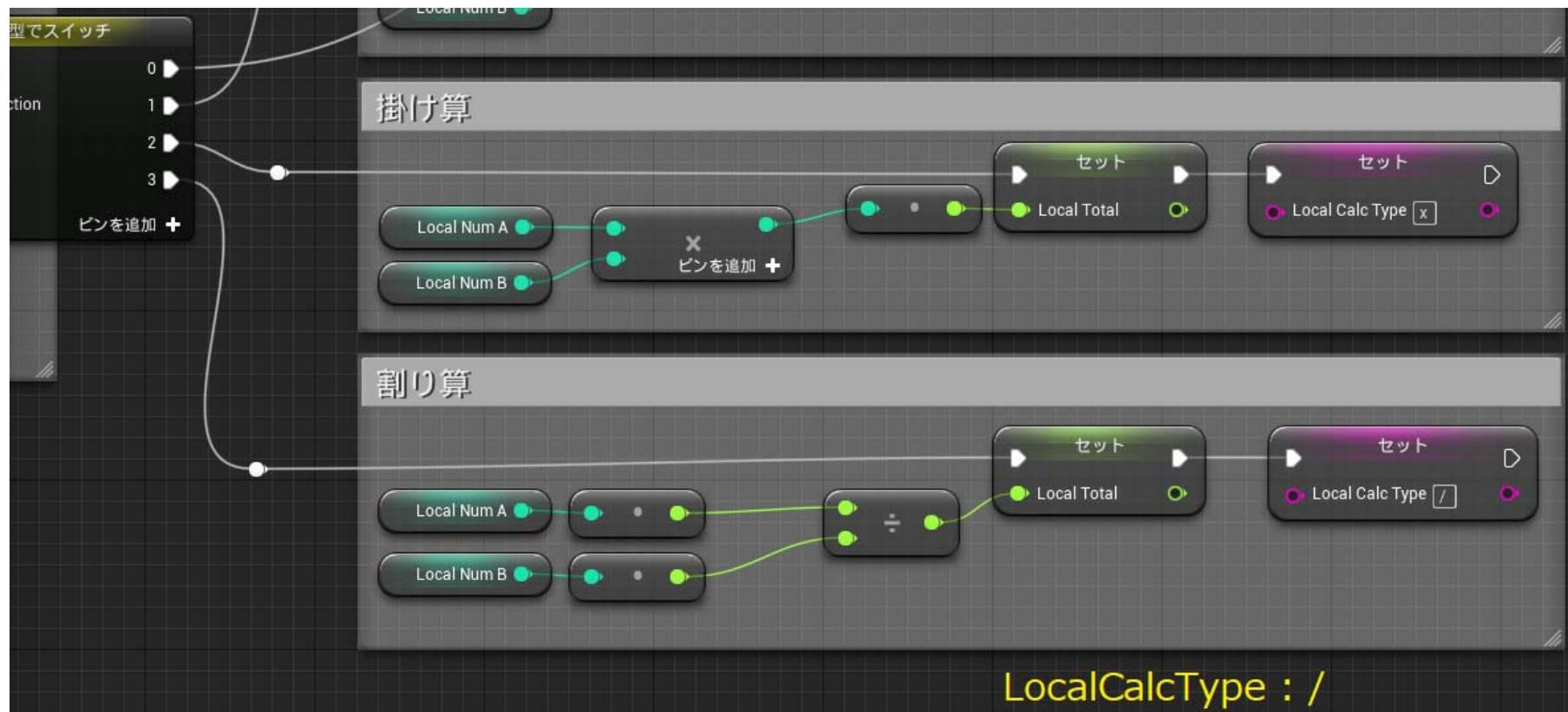
1の時に引き算



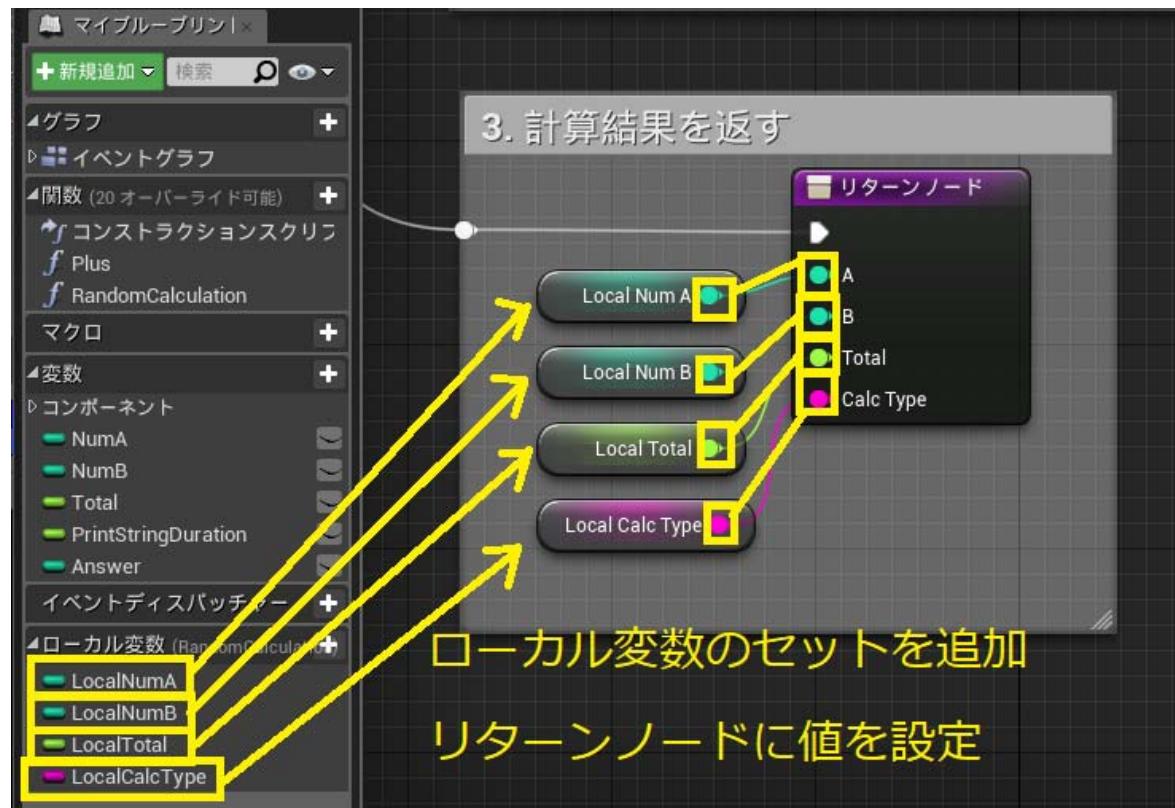
2の時に掛け算を実装



3の時に割り算を実装



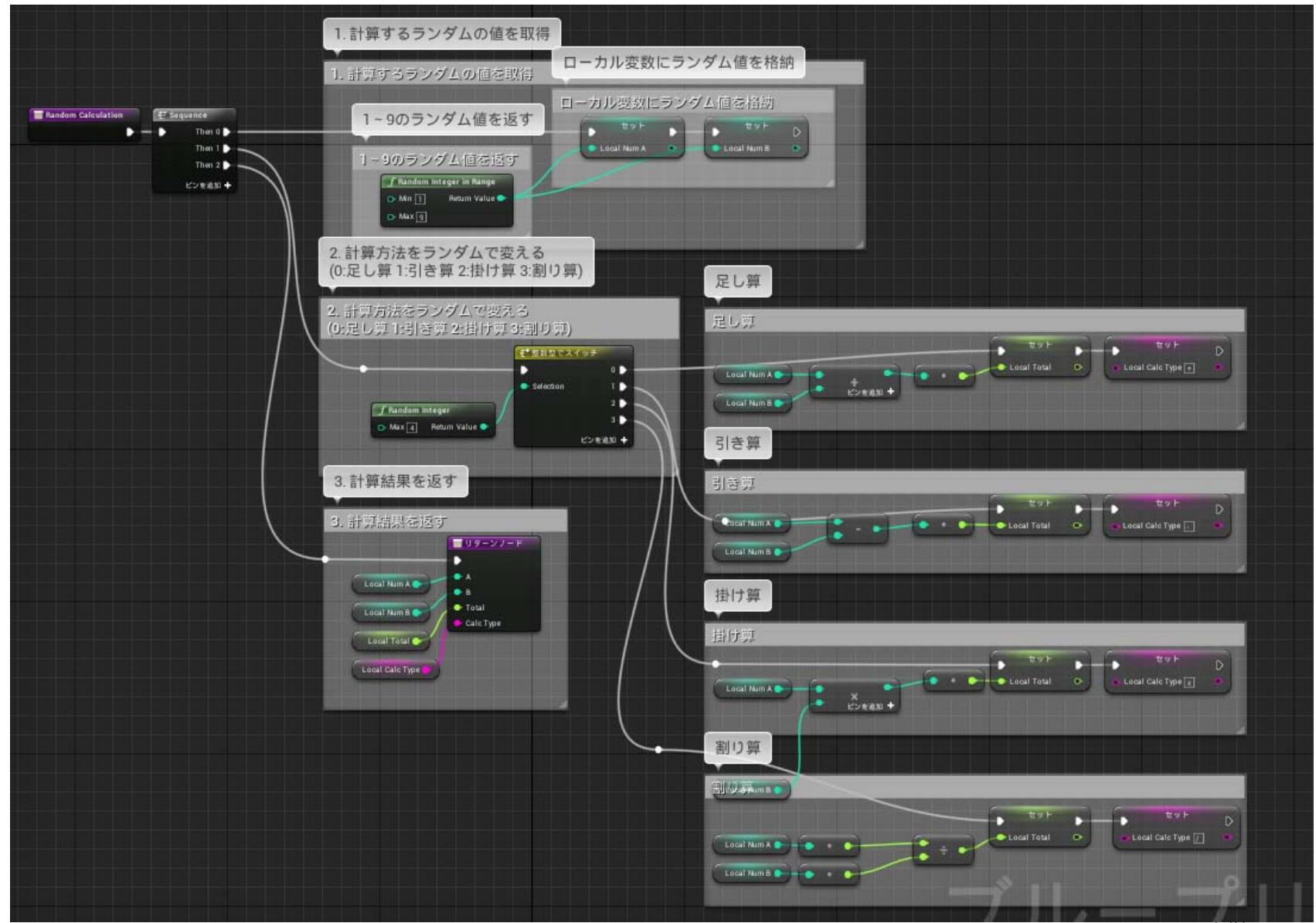
ローカル変数を リターンノードのパラメータにつなげる



ローカル変数のセットを追加

リターンノードのパラメータを設定

完成



5. 計算方法(足し算、引き算、掛け算、割り算)をランダムに修正する

5.1 新規レベル (RandomCalcGame) を作成・保存

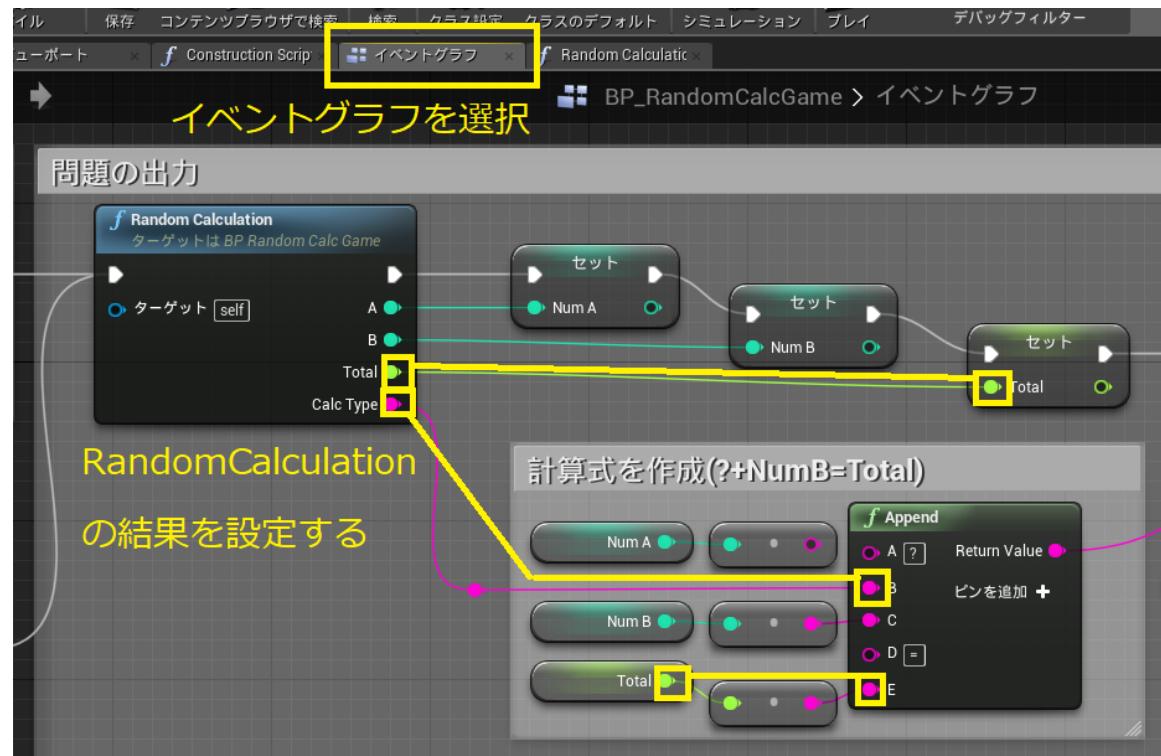
5.2 BP_RandomPlusGameを複製して、BP_RandomCalcGameを作成

5.3 引き算、掛け算、割り算を実装

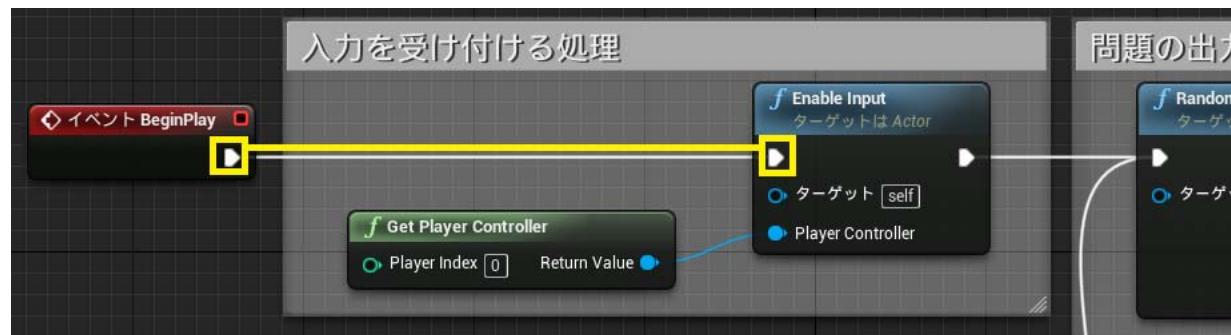
5.4 関数RandomCalculationの修正

5.5 算数ゲームの計算方法のランダム対応

RandomCalculationの結果を設定する



イベントBeginPlayの接続を元に戻して、
プレイして確認



イベントBeginPlayの接続を元に戻す



?+7=15.0 足し算
True
3
?x9=27.0 掛け算
True
4
?/1=4.0 割り算

プレイして確認
ランダムで計算が行われる

6. UMG（ウィジェットブループリント）の 作成・表示

6. UMGの作成・表示（ウィジェットブループリント）

6.1 新規レベル（DispWidget）を作成・保存

6.2 画像ファイルのインポート

6.3 ウィジェットブループリント(WBP_MathGameOutline)を作成

6.4 ウィジェットブループリントエディタについて

6.5 ウィジェットの配置

 6.5.1 Imageを使用してアウトライン背景を表示

 6.5.2 Textを配置して計算式を作成

 6.5.3 HorizonBoxを使用して、1~9のButtonを横並びに配置する

 6.5.4 Imageを使用して結果判定の画像を配置

6.6 レベルブループリントの編集

 6.6.1 ウィジェットを表示

 6.6.2 マウスカーソルを常に表示

 6.6.3 ESCキーでゲームを終了

6. UMGの作成・表示（ウィジェットブループリント）

6.1 新規レベル（DispWidget）を作成・保存

6.2 画像ファイルのインポート

6.3 ウィジェットブループリント(WBP_MathGameOutline)を作成

6.4 ウィジェットブループリントエディタについて

6.5 ウィジェットの配置

6.5.1 Imageを使用してアウトライン背景を表示

6.5.2 Textを配置して計算式を作成

6.5.3 HorizonBoxを使用して、1~9のButtonを横並びに配置する

6.5.4 Imageを使用して結果判定の画像を配置

6.6 レベルブループリントの編集

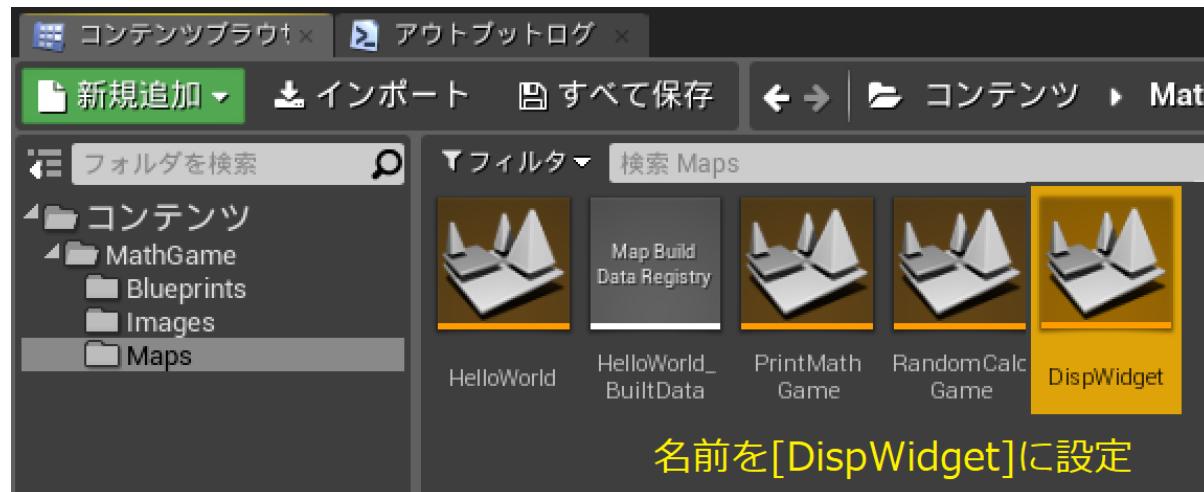
6.6.1 ウィジェットを表示

6.6.2 マウスカーソルを常に表示

6.6.3 ESCキーでゲームを終了

新規レベル (DispWidget) の作成

1. ファイル > 新規レベル(Ctrl+N)
2. [空のレベル]を選択する
3. ファイル > 現在のレベルを保存(Ctrl+S)
4. Mapsフォルダに、名前を[DispWidget]に設定して保存



6. UMGの作成・表示（ウィジェットブループリント）

6.1 新規レベル（DispWidget）を作成・保存

6.2 画像ファイルのインポート

6.3 ウィジェットブループリント(WBP_MathGameOutline)を作成

6.4 ウィジェットブループリントエディタについて

6.5 ウィジェットの配置

 6.5.1 Imageを使用してアウトライン背景を表示

 6.5.2 Textを配置して計算式を作成

 6.5.3 HorizonBoxを使用して、1~9のButtonを横並びに配置する

 6.5.4 Imageを使用して結果判定の画像を配置

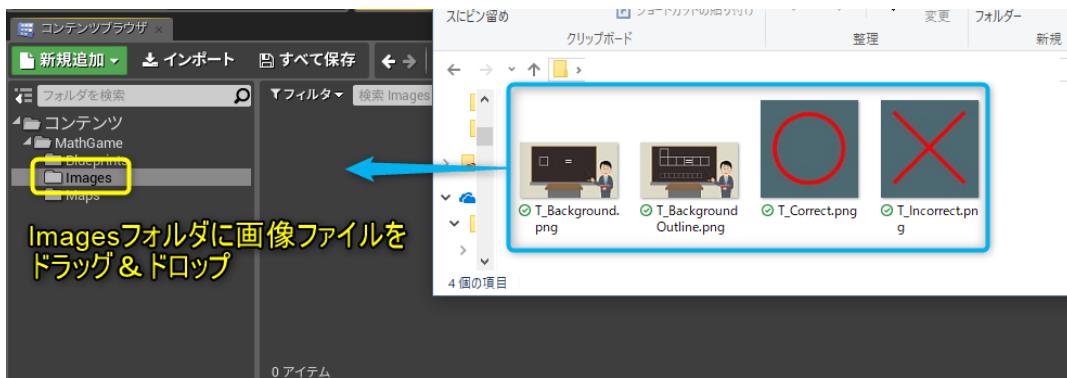
6.6 レベルブループリントの編集

 6.6.1 ウィジェットを表示

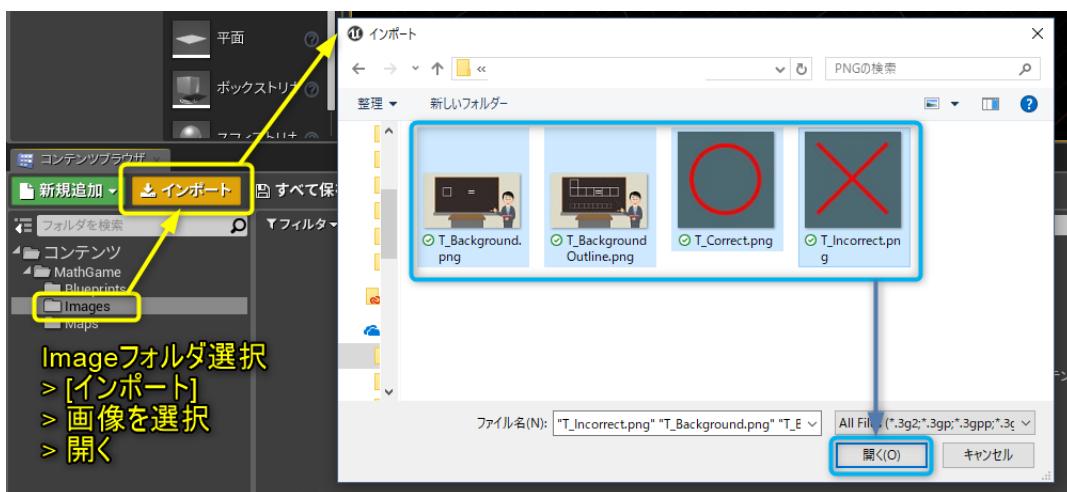
 6.6.2 マウスカーソルを常に表示

 6.6.3 ESCキーでゲームを終了

画像のインポート

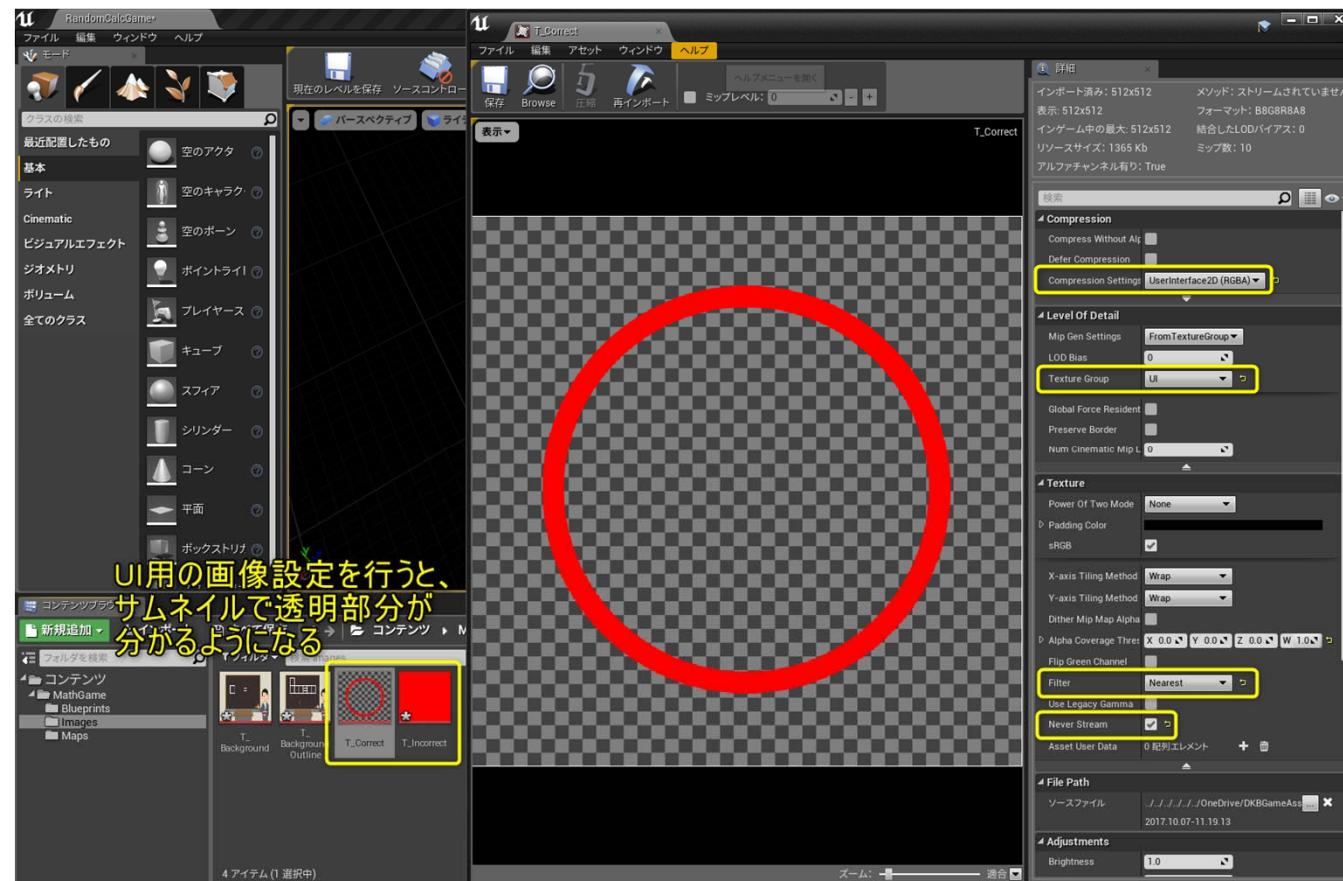


エクスプローラの画像ファイルを
UE4のエディタ Imagesフォルダに
ドラッグ & ドロップ



コンテンツブラウザ[インポート]をクリック
> 画像ファイルを選択 > [開く]

インポートした画像の設定



パラメータ名	設定
Compression Settings	UserInterface2D(RGBA)
TextureGroup	UI
Filter	Nearest
Never Stream	チェック

UI用の画像設定について

UE4 UI用画像の設定について

<http://denshikousakubu.com/2017/10/24/20171024 UE4 UM GTextureSettings/>

6. UMGの作成・表示（ウィジェットブループリント）

6.1 新規レベル（DispWidget）を作成・保存

6.2 画像ファイルのインポート

6.3 ウィジェットブループリント(WBP_MathGameOutline)を作成

6.4 ウィジェットブループリントエディタについて

6.5 ウィジェットの配置

 6.5.1 Imageを使用してアウトライン背景を表示

 6.5.2 Textを配置して計算式を作成

 6.5.3 HorizonBoxを使用して、1~9のButtonを横並びに配置する

 6.5.4 Imageを使用して結果判定の画像を配置

6.6 レベルブループリントの編集

 6.6.1 ウィジェットを表示

 6.6.2 マウスカーソルを常に表示

 6.6.3 ESCキーでゲームを終了

ウィジェットブループリント (WBP_MathGameOutline)を作成



Blueprints フォルダを選択
右クリック > 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

6. UMGの作成・表示（ウィジェットブループリント）

6.1 新規レベル（DispWidget）を作成・保存

6.2 画像ファイルのインポート

6.3 ウィジェットブループリント(WBP_MathGameOutline)を作成

6.4 ウィジェットブループリントエディタについて

6.5 ウィジェットの配置

 6.5.1 Imageを使用してアウトライン背景を表示

 6.5.2 Textを配置して計算式を作成

 6.5.3 HorizonBoxを使用して、1~9のButtonを横並びに配置する

 6.5.4 Imageを使用して結果判定の画像を配置

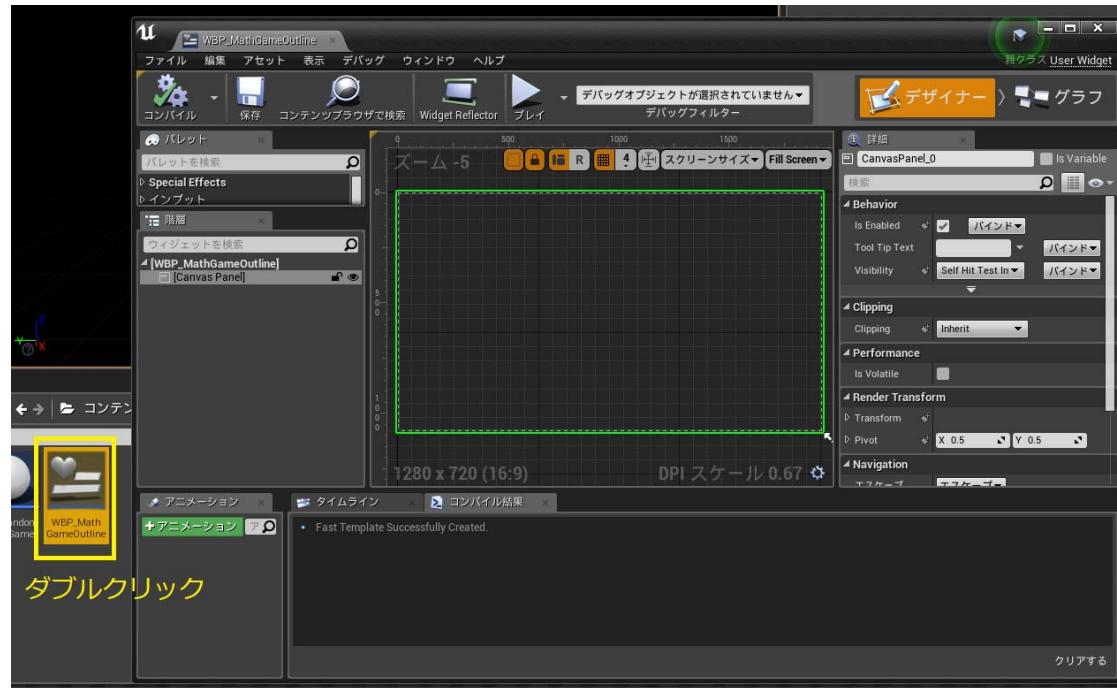
6.6 レベルブループリントの編集

 6.6.1 ウィジェットを表示

 6.6.2 マウスカーソルを常に表示

 6.6.3 ESCキーでゲームを終了

ウィジェットブループリントエディタについて



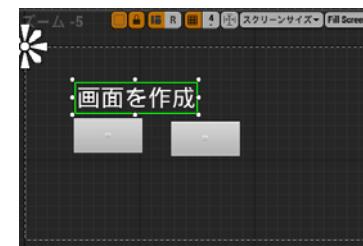
ウィジェットブループリントエディタの詳細

<https://docs.unrealengine.com/latest/JPN/Engine/UMG/UserGuide/WidgetBlueprints/index.html>

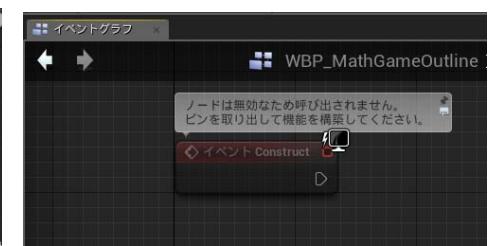


[右上]

デザイナー,グラフ
を切り替え

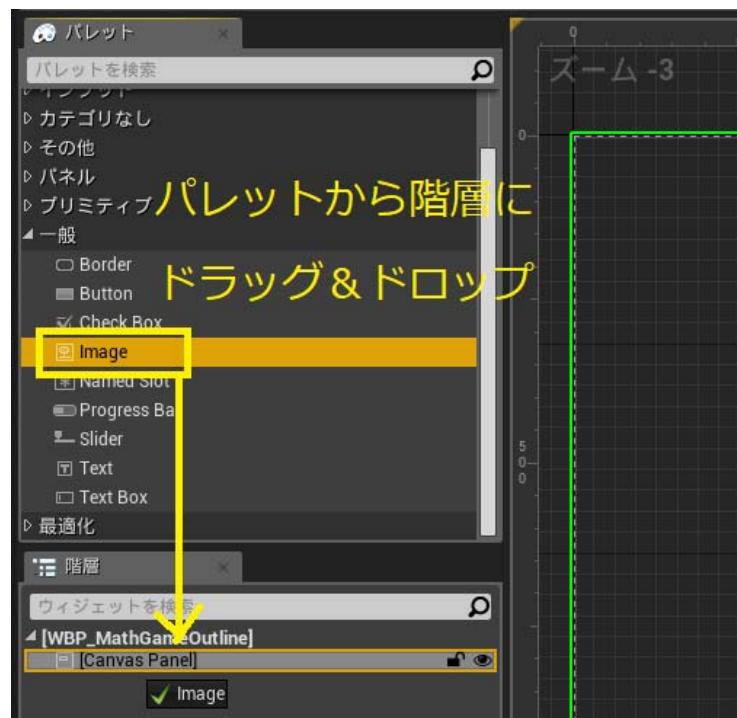


デザイナー
UIを作成

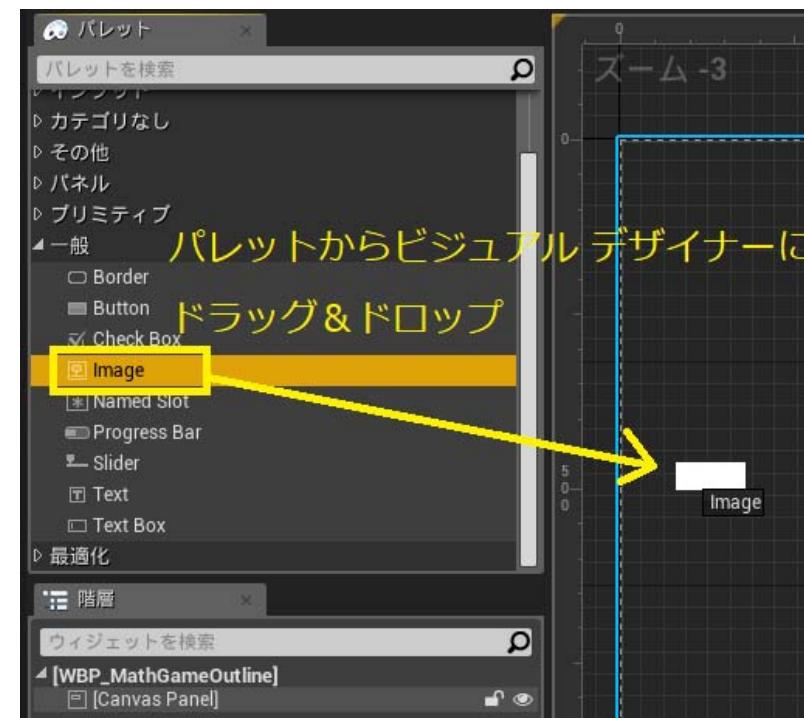


グラフ
UIの処理を作成

ウィジェットの追加

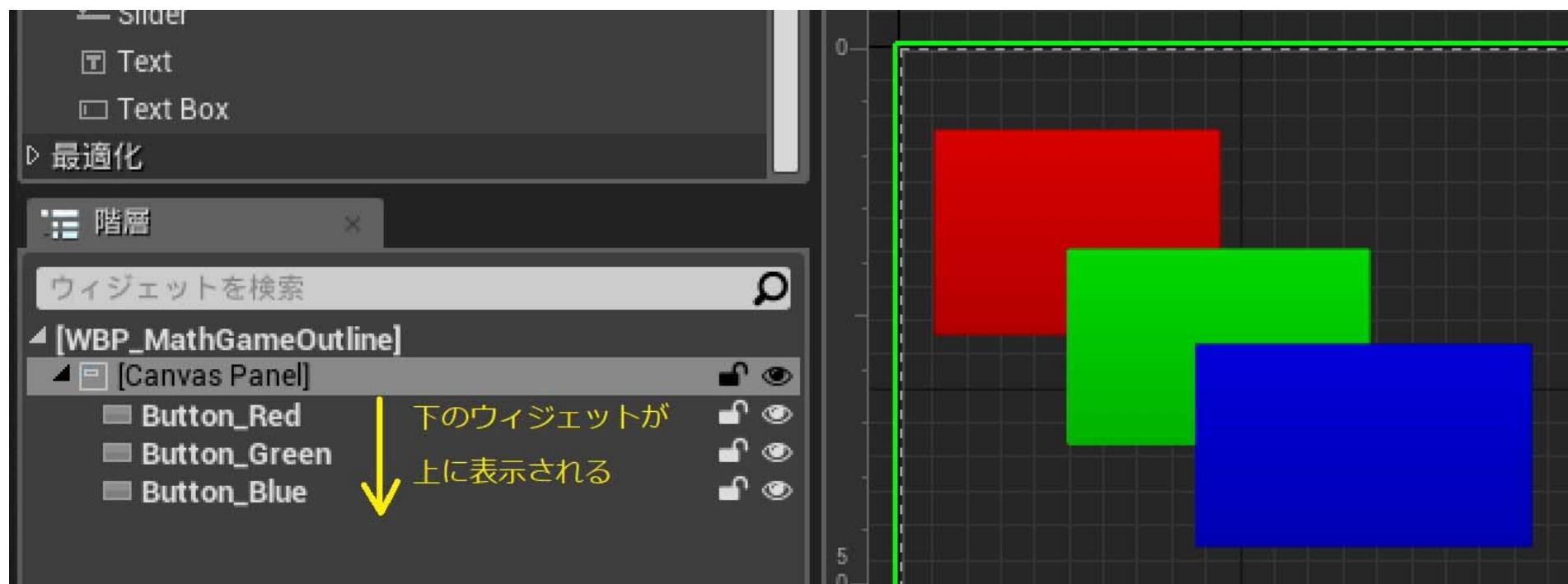


パレットからウィジェットを
階層タブにドラッグ&ドロップ



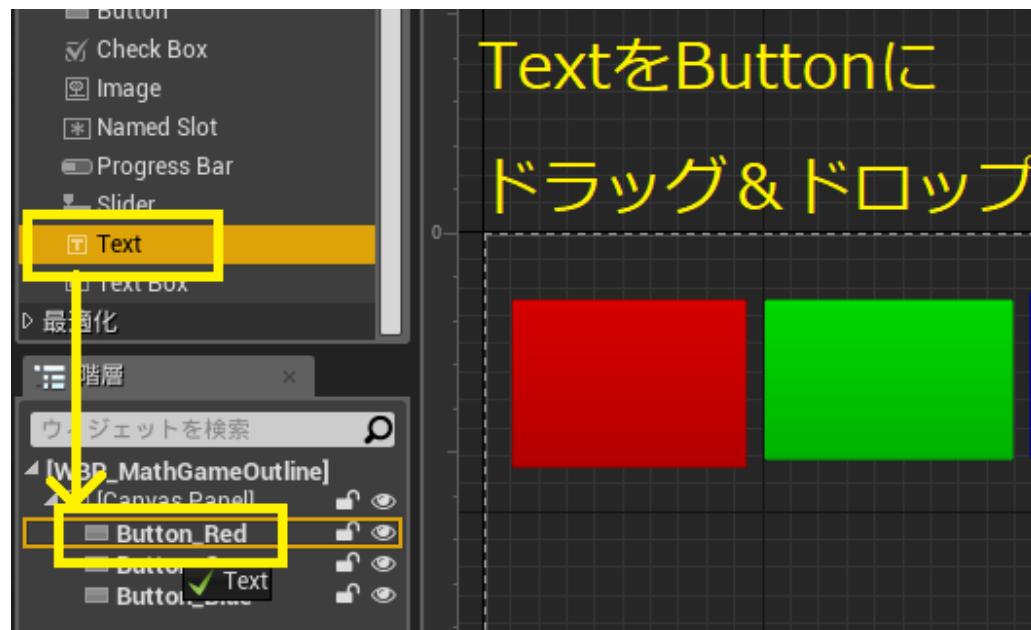
パレットからウィジェットを
ビジュアルデザイナーにドラッグ&ドロップ

階層タブで表示される ウィジェットの順番を設定

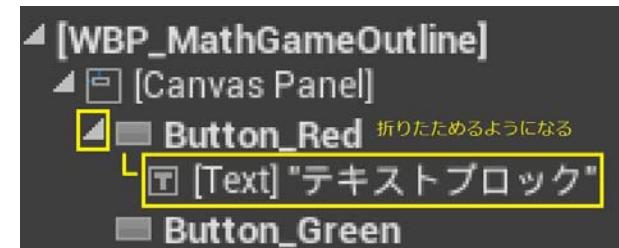


階層タブで表示される
ウィジェットの順番を決める
下の
ウィジェットが上に表示される

ウィジェットの親子関係 テキストを含むボタン



TextをButtonにドラッグ&ドロップ
(ButtonにTextを子として追加すると、
Textのあるボタンを作成することができる)

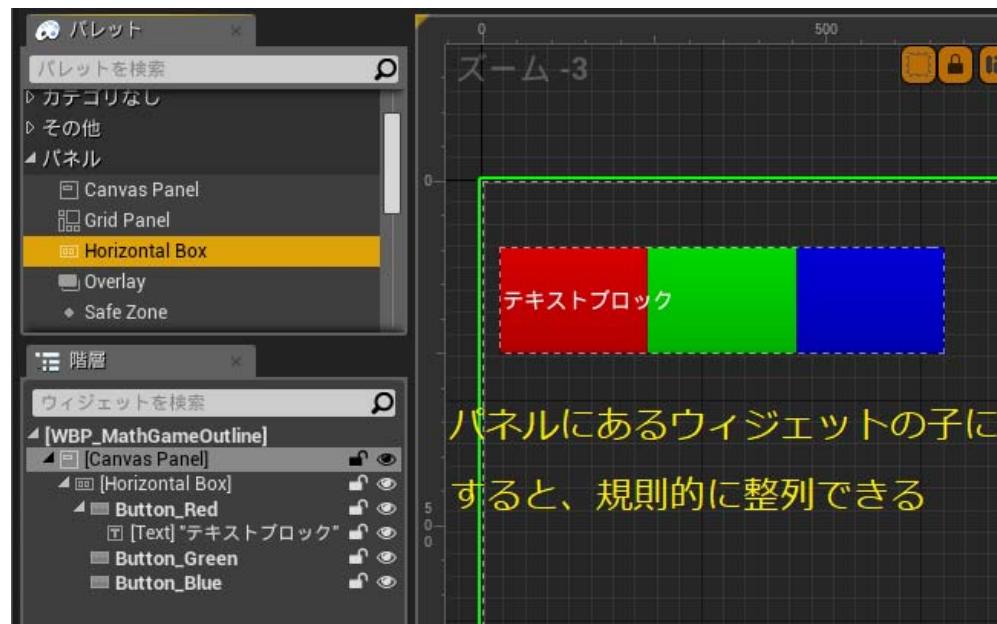


子が増えると折りたためる
ようになる

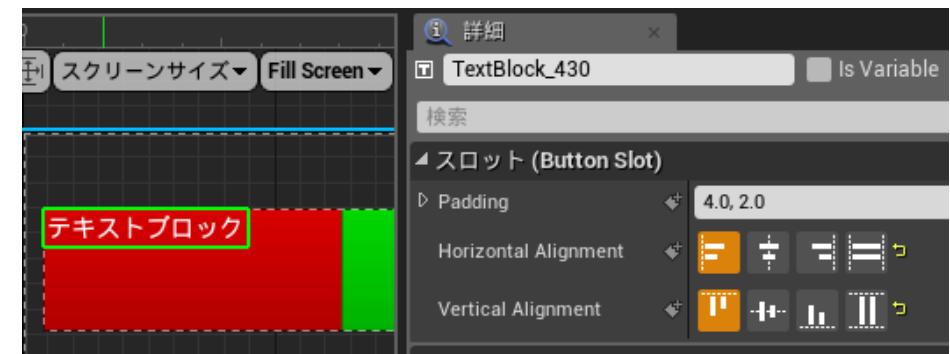


親のボタンを動かすと
子のテキストも一緒に動く

ウィジェットの子にすると 配置方法が変わる

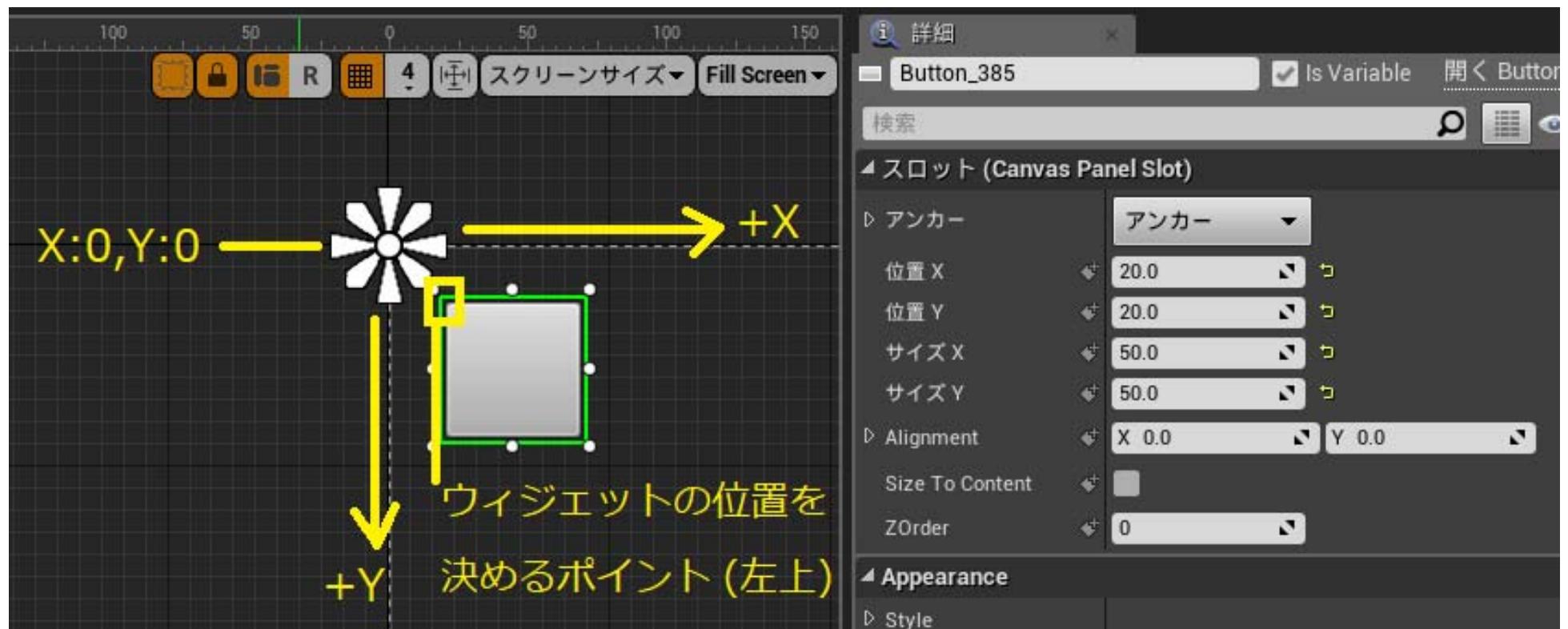


パネルにある項目の子供にすると、
横や縦に規則的に配置することができる
(Horizontal Box : 横方向に規則的に配置することができる)

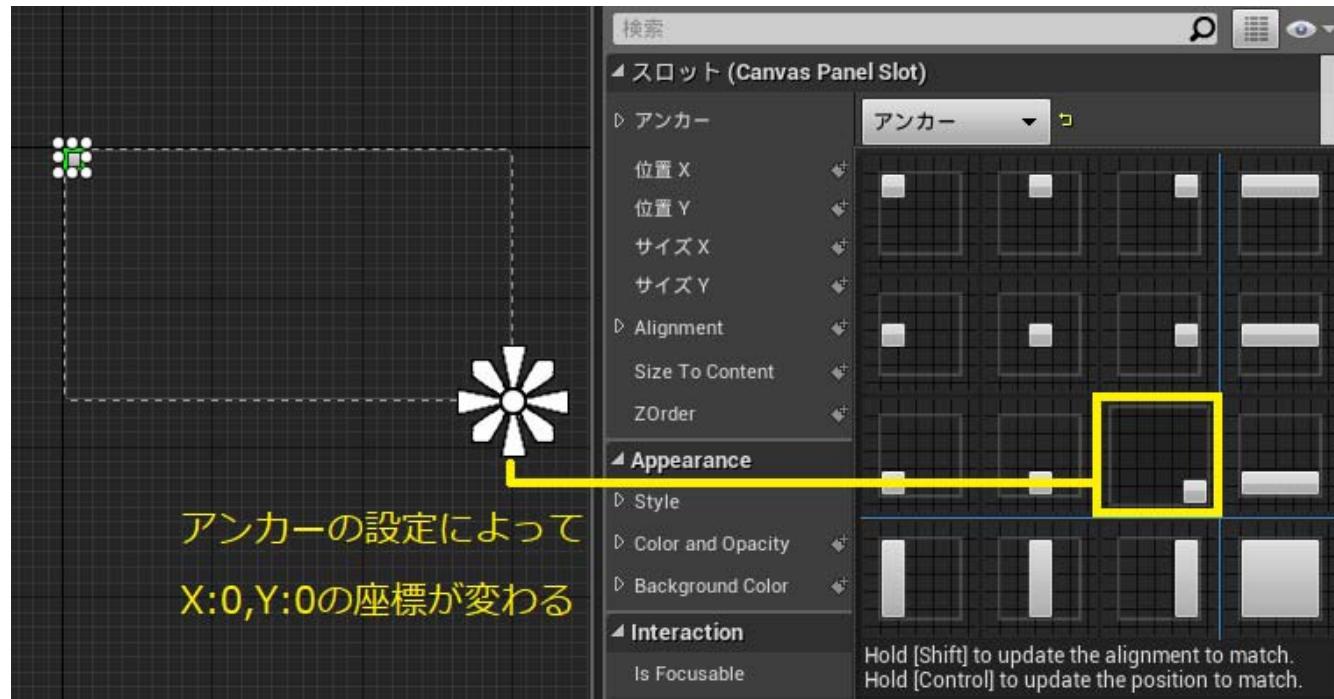


ボタンに配置したテキストも
上下や左右の揃え方も設定することができる

ウィジェットの座表



アンカーに設定によって、X:0,Y:0の位置が変わる

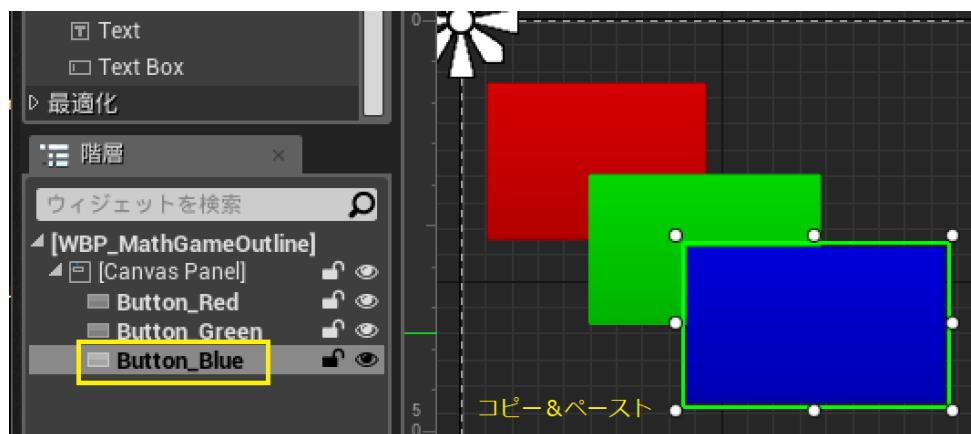


アンカー

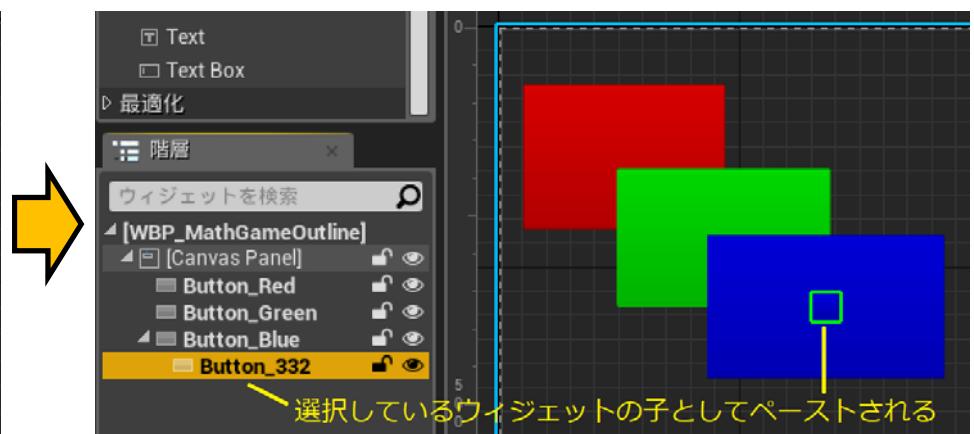
<https://docs.unrealengine.com/latest/JPN/Engine/UMG/UserGuide/Anchors/index.html>

ウィジェットのコピー & ペーストの注意

ウィジェットのコピー & ペーストは少し癖があるので注意

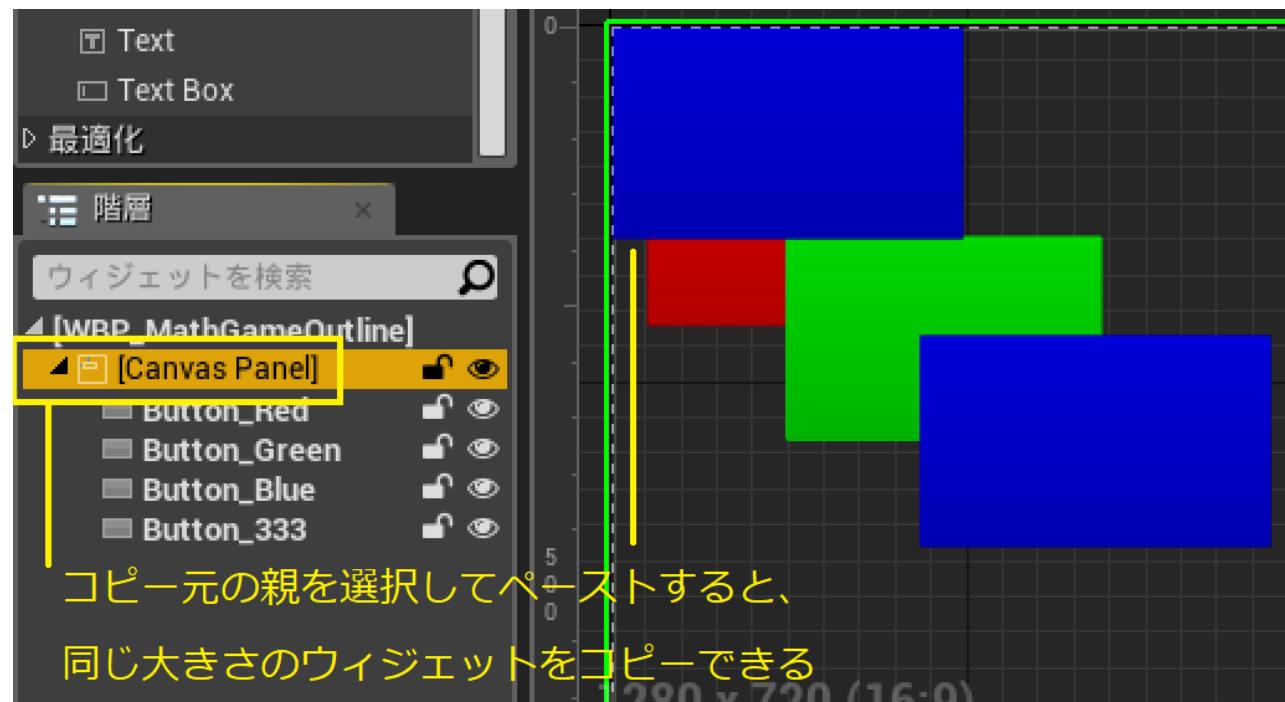


ウィジェットを選択してコピー & ペースト



選択しているウィジェットの子として
複製されるので、同じ大きさにならない

同じ大きさでコピー＆ペーストする際には
コピー元の親を選択して、ペースト



コピー元の親を選択してペースト
ウィジェットが同一階層に複製されるので、同じ大きさになる

6. UMGの作成・表示（ウィジェットブループリント）

6.1 新規レベル（DispWidget）を作成・保存

6.2 画像ファイルのインポート

6.3 ウィジェットブループリント(WBP_MathGameOutline)を作成

6.4 ウィジェットブループリントエディタについて

6.5 ウィジェットの配置

6.5.1 Imageを使用してアウトライン背景を表示

6.5.2 Textを配置して計算式を作成

6.5.3 HorizonBoxを使用して、1~9のButtonを横並びに配置する

6.5.4 Imageを使用して結果判定の画像を配置

6.6 レベルブループリントの編集

6.6.1 ウィジェットを表示

6.6.2 マウスカーソルを常に表示

6.6.3 ESCキーでゲームを終了

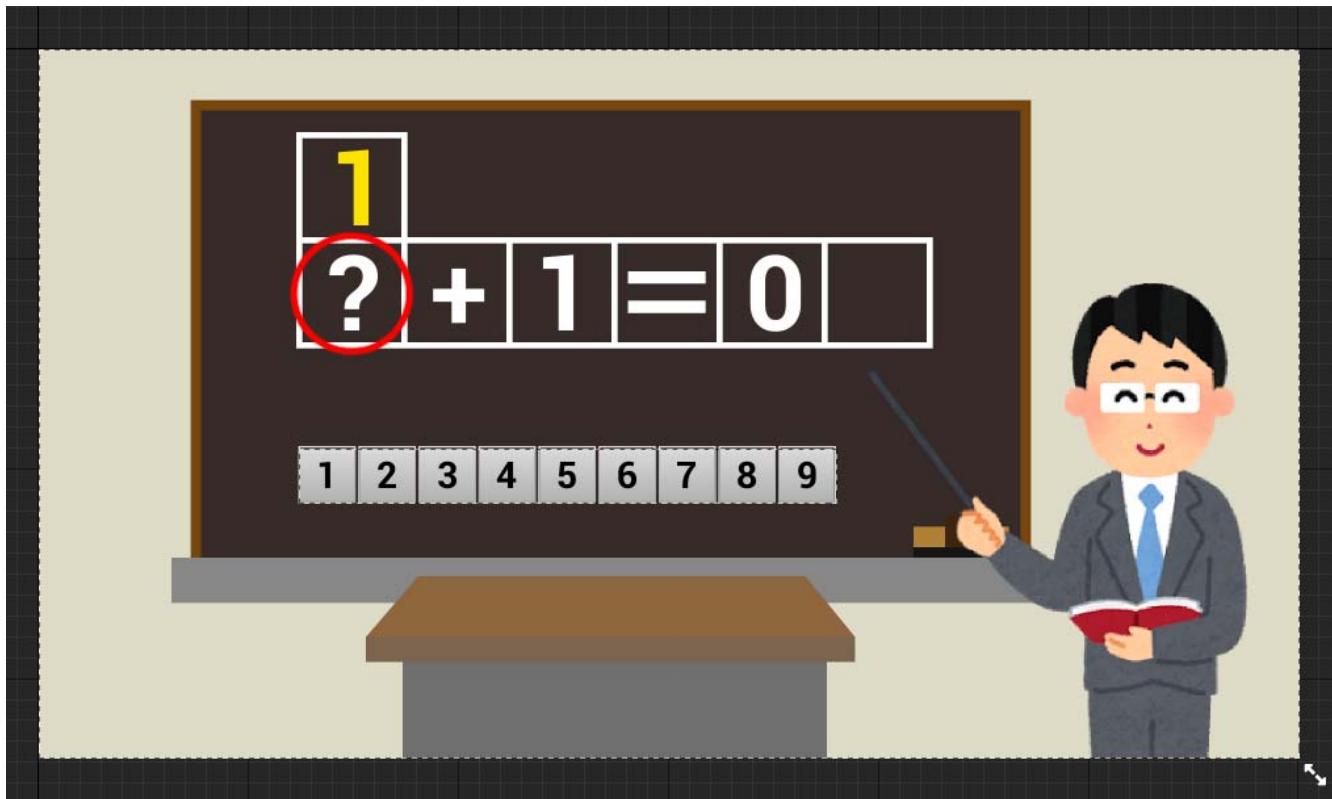
今回使用するウィジェット

説明
ボタンの処理を実装
スレートブラシ、テクスチャ、スプライト、またはマテリアルを表示
画面上にテキストを表示 オプションや他の UI エレメントのテキストを記述するために使用
フローで子ウィジェットを横方向にレイアウト

Widget タイプのリファレンス

<https://docs.unrealengine.com/latest/JPN/Engine/UMG/UserGuide/WidgetTypeReference/index.html>

必要なウィジェットを配置してUMGを作成



6. UMGの作成・表示（ウィジェットブループリント）

6.1 新規レベル（DispWidget）を作成・保存

6.2 画像ファイルのインポート

6.3 ウィジェットブループリント(WBP_MathGameOutline)を作成

6.4 ウィジェットブループリントエディタについて

6.5 ウィジェットの配置

 6.5.1 Imageを使用してアウトライン背景を表示

 6.5.2 Textを配置して計算式を作成

 6.5.3 HorizonBoxを使用して、1~9のButtonを横並びに配置する

 6.5.4 Imageを使用して結果判定の画像を配置

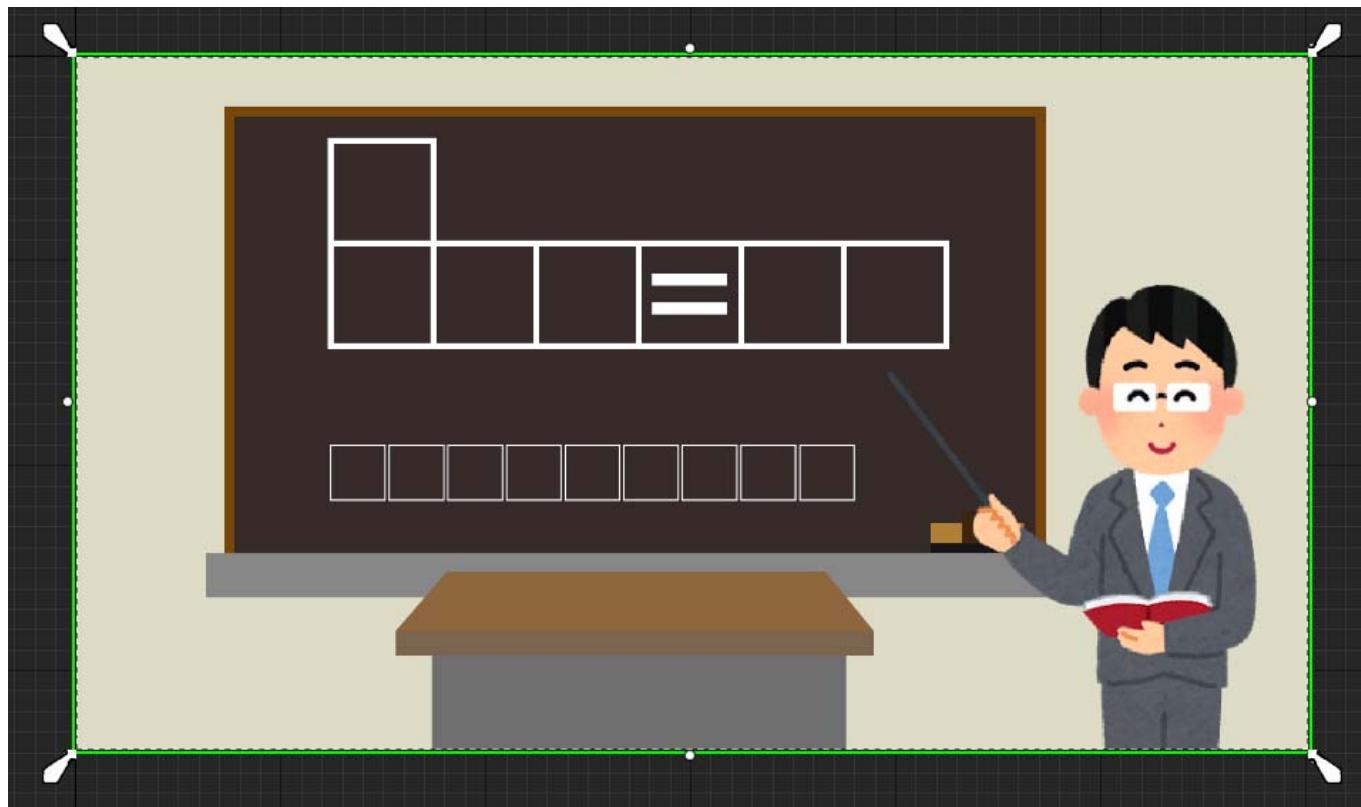
6.6 レベルブループリントの編集

 6.6.1 ウィジェットを表示

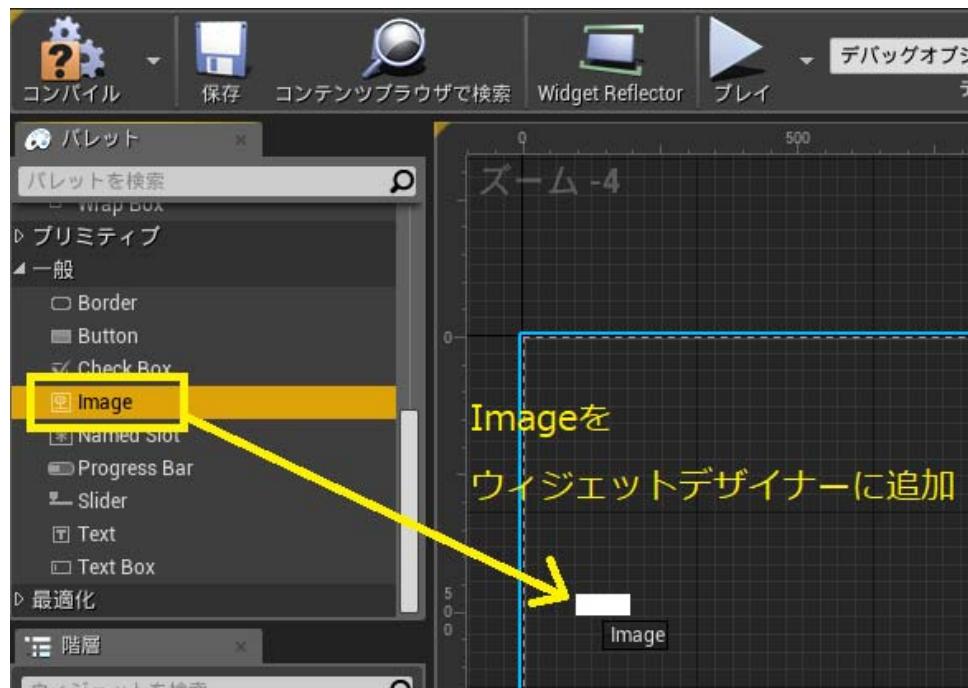
 6.6.2 マウスカーソルを常に表示

 6.6.3 ESCキーでゲームを終了

ウィジェットを配置する場所のアウトライン
が書かれた画像を表示する

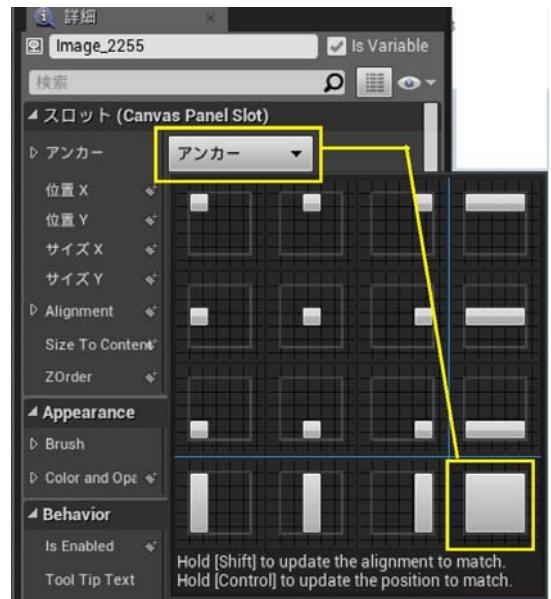


Imageをウィジェットデザイナーに追加

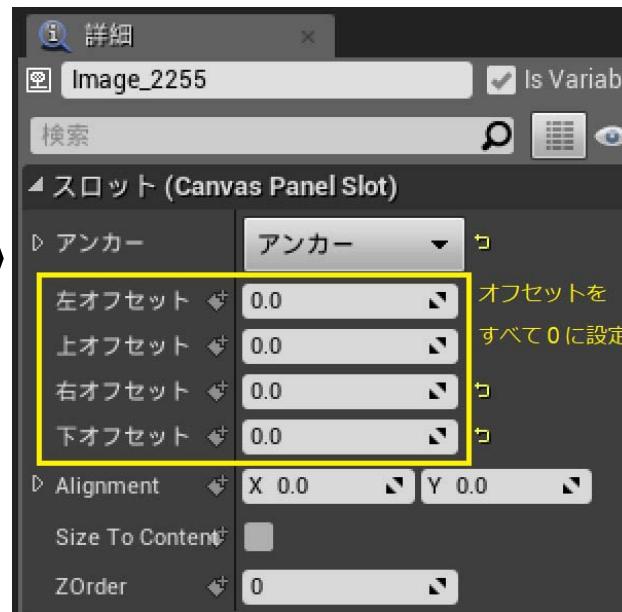


パレットタブ > 一般 > Imageを
ウィジェットデザイナーにドロップ

Canvas_Panelに隙間なく 画像が表示されるように設定する



アンカーをクリック
> 右下のアイコンをクリック

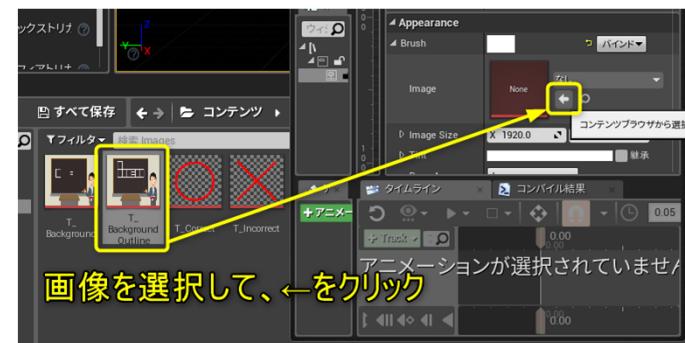
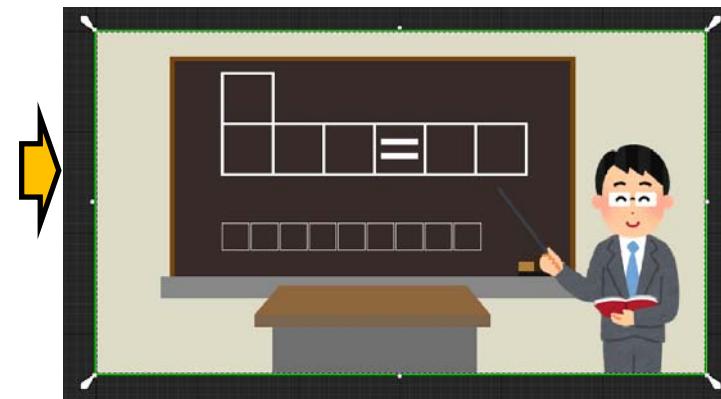
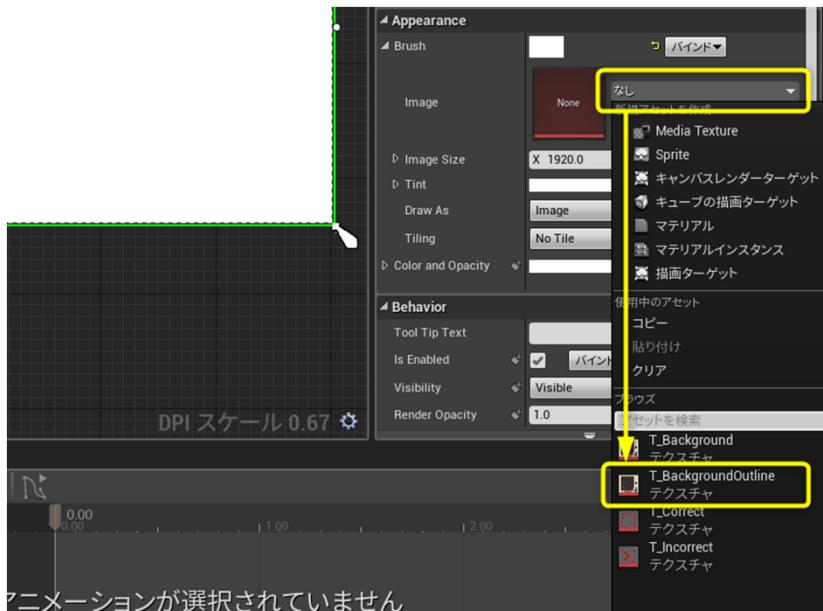


オフセットをすべて0に設定



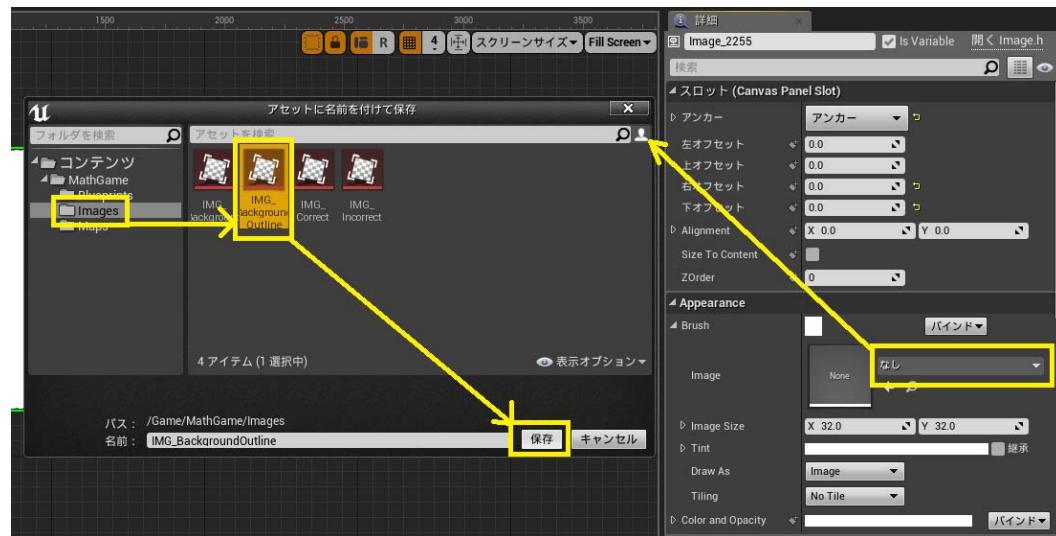
Canvas_Panelに隙間なく拡がる

BrushのImageに [T_BackgroundOutline]を設定

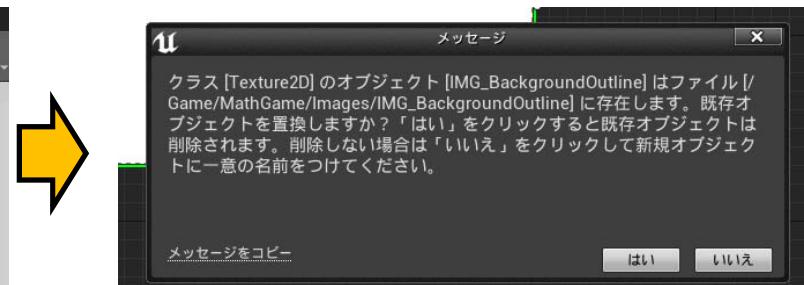


画像を選択してから、Brush > Imageの←をクリック
(リストから画像を選択しない方がいい)

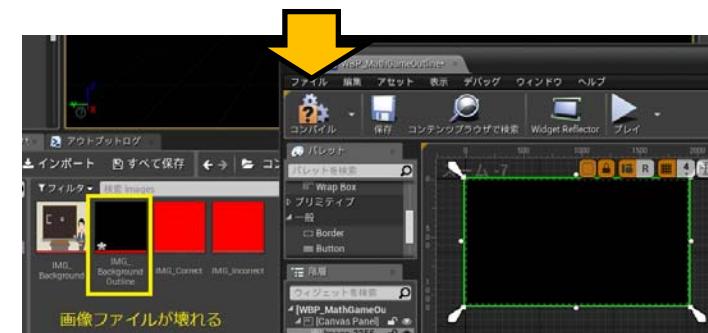
リストを選択した際にファイル選択のダイアログが出たときに、画像ファイルを選択するとファイルが壊れる



Brush > Image をクリックしたときに
ファイル選択ダイアログが表示される時がある（時々）
ダイアログから画像を選択する



メッセージダイアログで[はい]を選択すると
ファイルが削除される



黒いファイルを削除して、
再度画像をインポートする

6. UMGの作成・表示（ウィジェットブループリント）

6.1 新規レベル（DispWidget）を作成・保存

6.2 画像ファイルのインポート

6.3 ウィジェットブループリント(WBP_MathGameOutline)を作成

6.4 ウィジェットブループリントエディタについて

6.5 ウィジェットの配置

6.5.1 Imageを使用してアウトライン背景を表示

6.5.2 Textを配置して計算式を作成

6.5.3 HorizonBoxを使用して、1~9のButtonを横並びに配置する

6.5.4 Imageを使用して結果判定の画像を配置

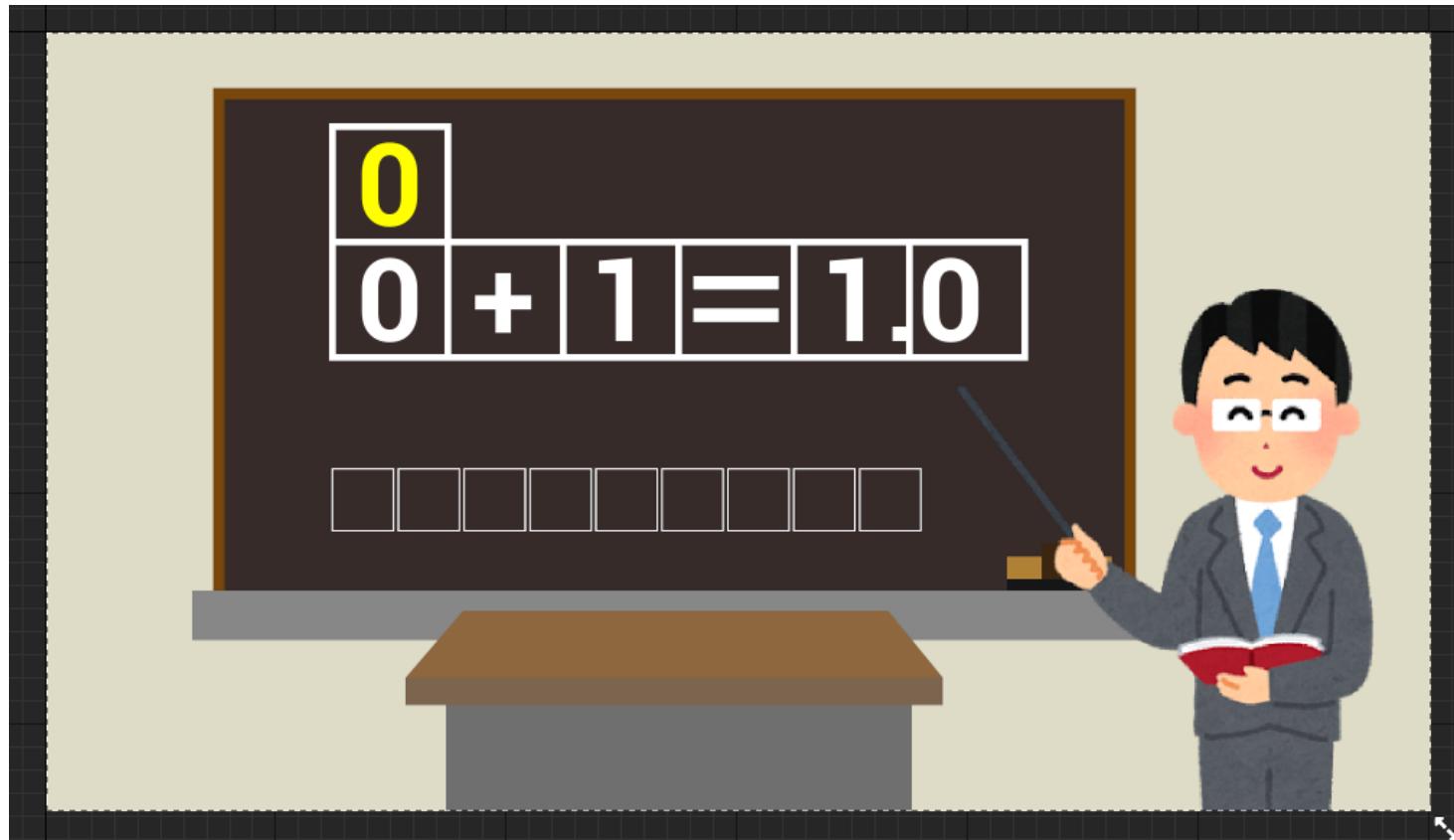
6.6 レベルブループリントの編集

6.6.1 ウィジェットを表示

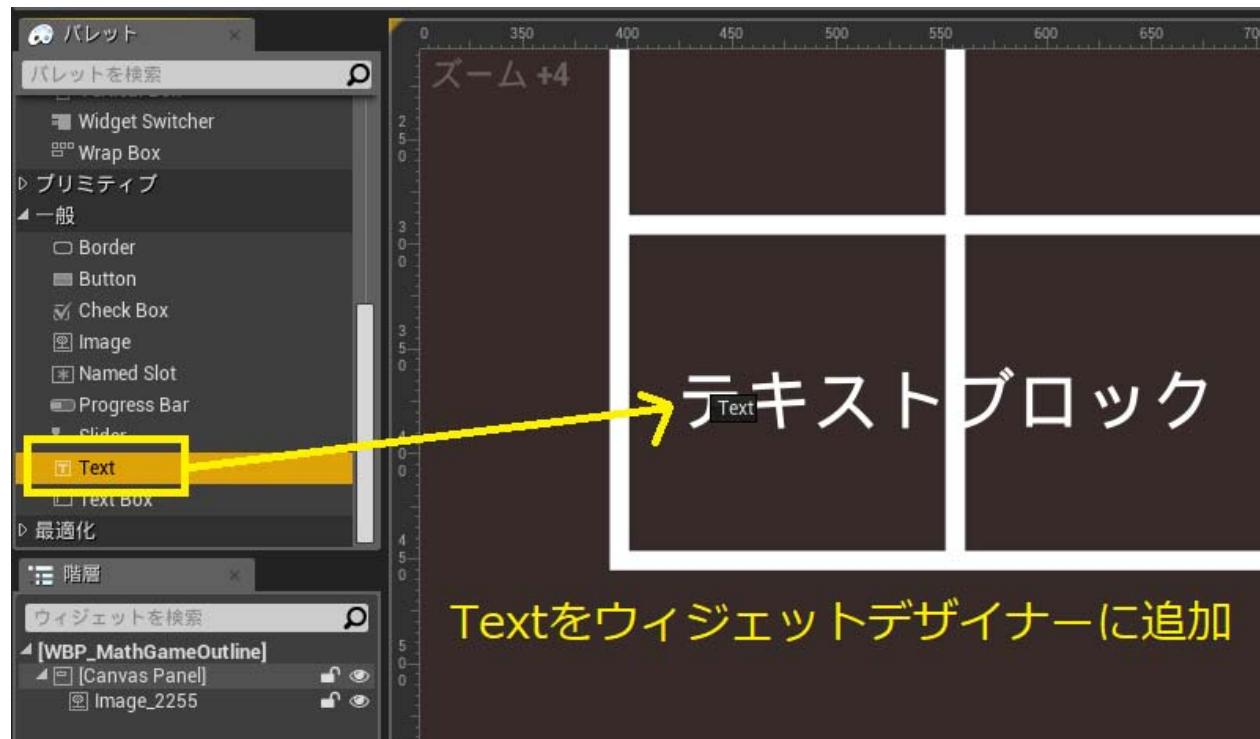
6.6.2 マウスカーソルを常に表示

6.6.3 ESCキーでゲームを終了

Textを配置して、計算式と答えを表示

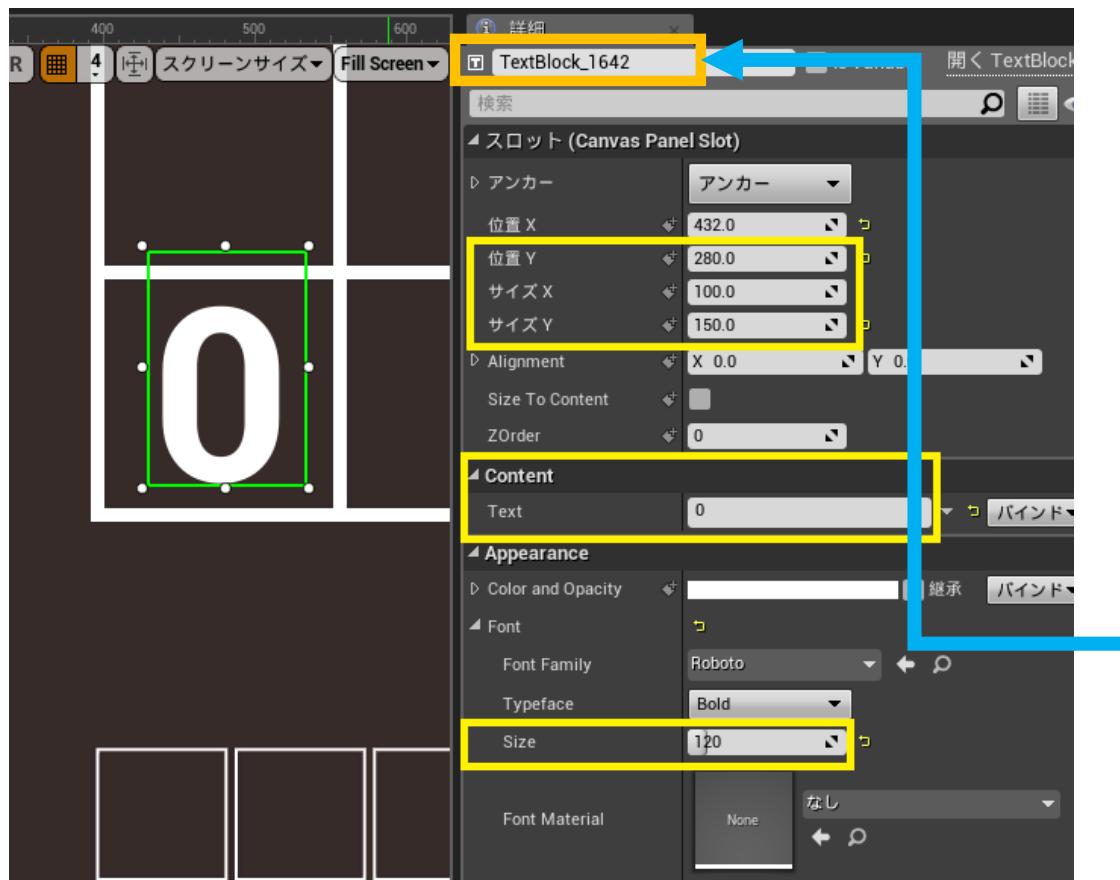


Textをウィジェットデザイナーに追加



パレットタブ > 一般 > Textを
ウィジェットデザイナーにドラッグ&ドロップ

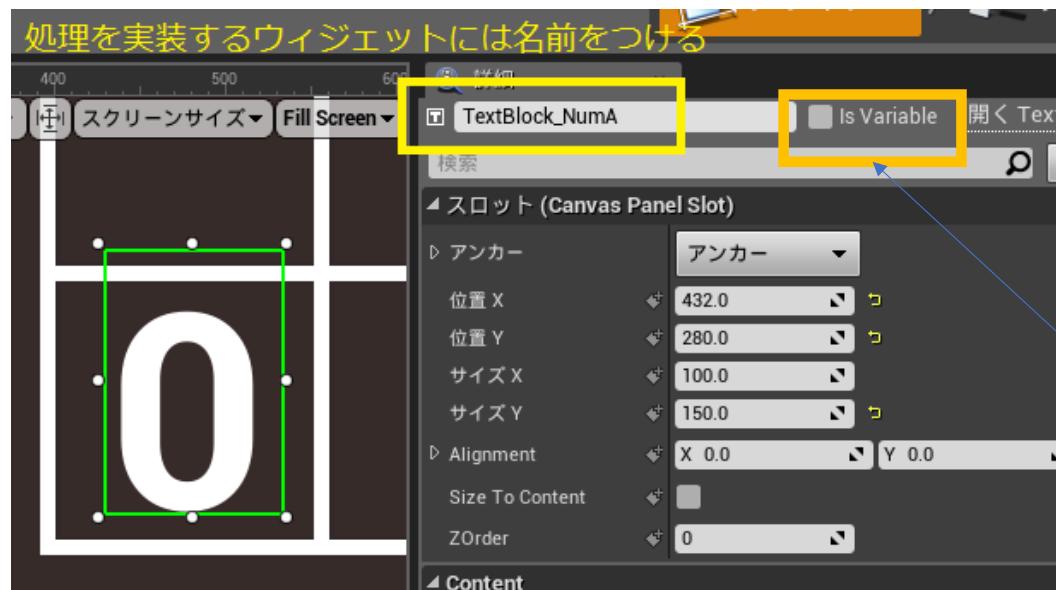
Textの詳細を設定



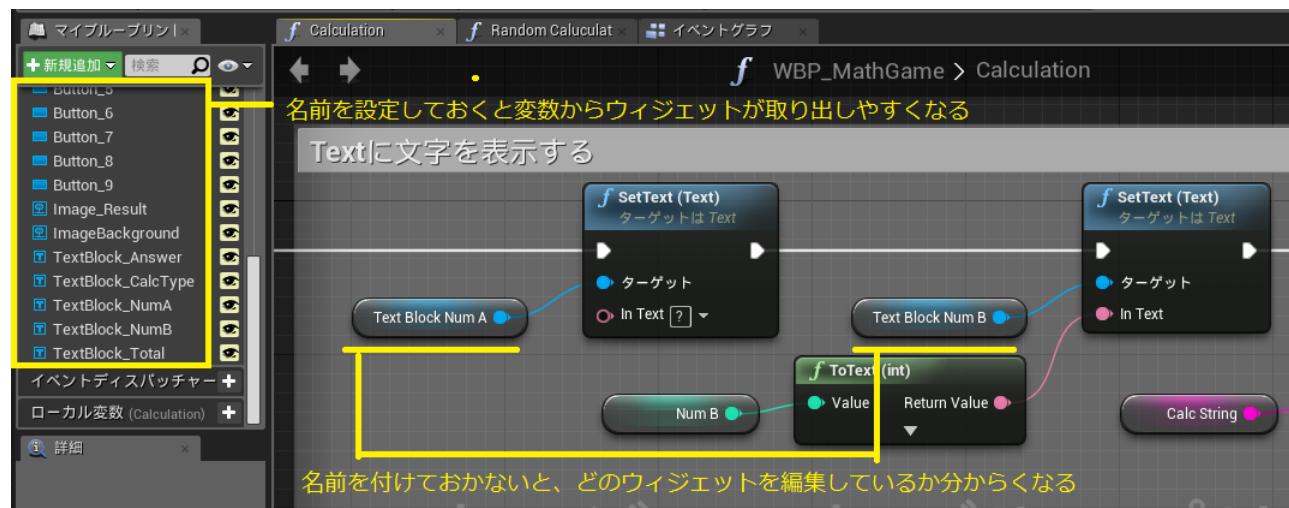
プロパティ	値
位置X	中心になるように設定
位置Y	280.0
サイズX	100.0
サイズY	150.0
Text	0
Font > Size	120

プロパティ	値
名前	TextBlock_NumA

値が変わるウィジェットには
名前を必ず決めて設定する



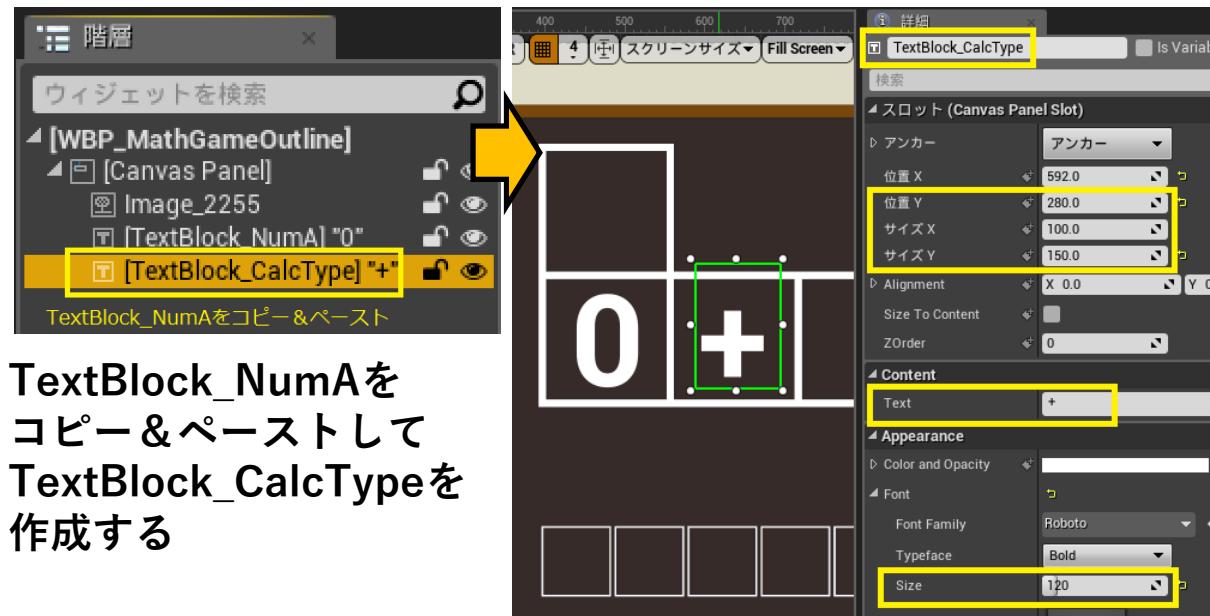
ウィジェットはウィジェットデザイナーに追加した際、名前には連番が設定される
処理を実装するウィジェットには名前を設定



IsVariableにチェックを入れると、
変数にウィジェット名が表示される

名前を付けておかないと、
どのウィジェットを処理するのか
わからなくなる

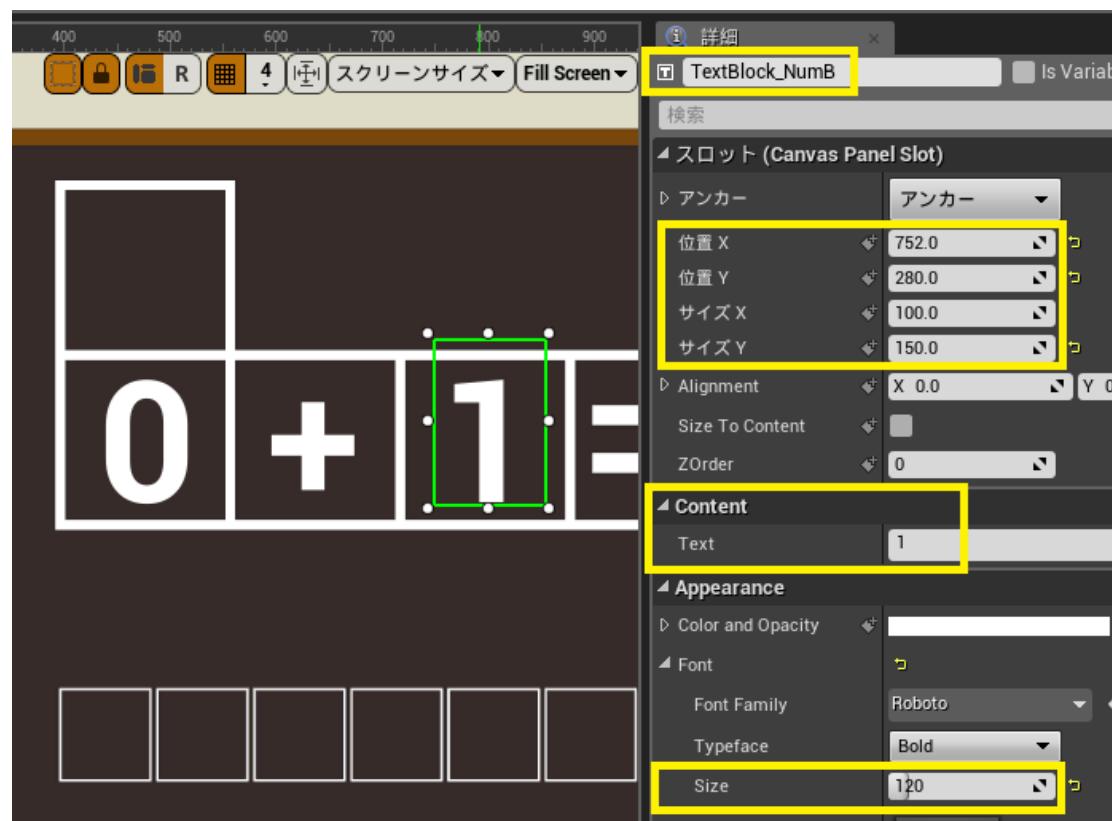
TextBlock_NumAをコピー&ペーストして、TextBlock_CalcTypeを作成



TextBlock_NumAを
コピー&ペーストして
TextBlock_CalcTypeを
作成する

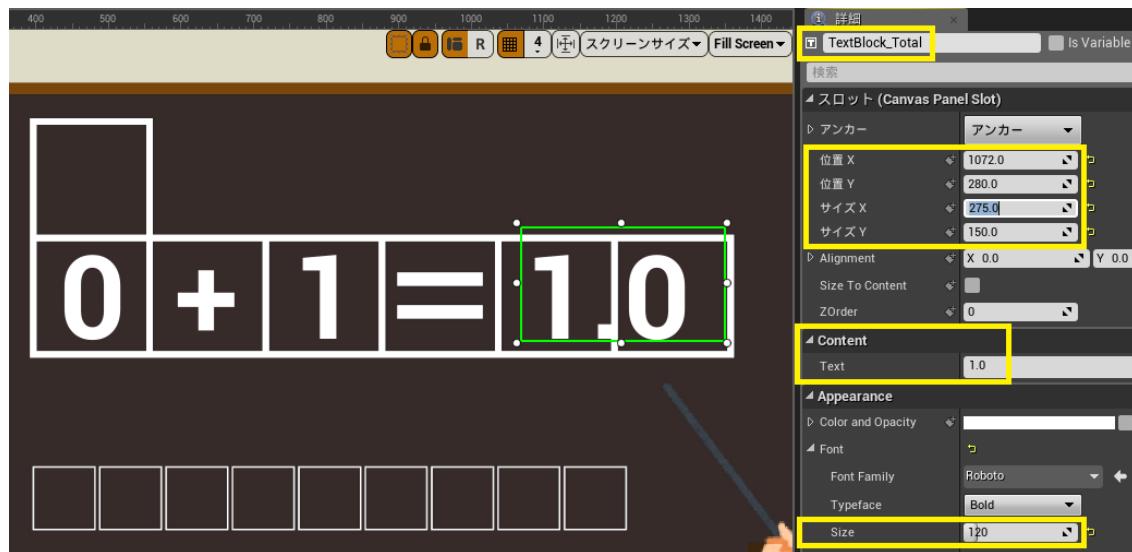
プロパティ	値
名前	TextBlock_CalcType
位置X	中心になるように設定
位置Y	280.0
サイズX	100.0
サイズY	150.0
Text	+
Font > Size	120

TextBlock_NumAをコピー&ペーストして、TextBlock_NumBを作成



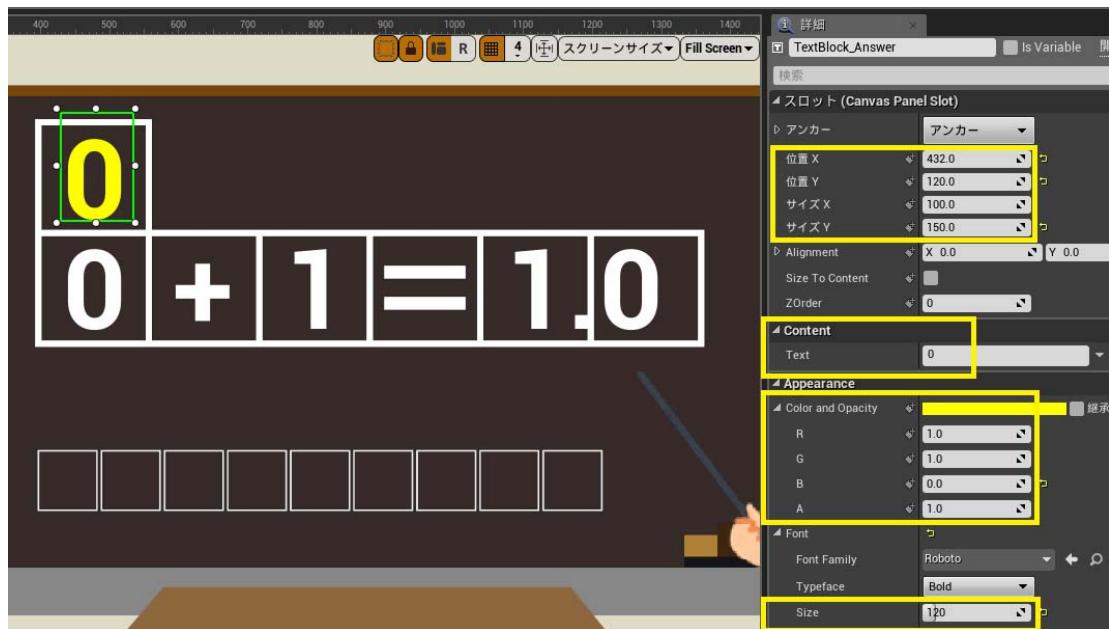
プロパティ	値
名前	TextBlock_NumB
位置X	中心になるように設定
位置Y	280.0
サイズX	100.0
サイズY	150.0
Text	1
Font > Size	120

TextBlock_NumAをコピー&ペーストして、
TextBlock_Totalを作成



プロパティ	値
名前	TextBlock_Total
位置X	中心になるように設定
位置Y	280.0
サイズX	275.0
サイズY	150.0
Text	1.0
Font > Size	120

TextBlock_NumAをコピー&ペーストして、TextBlock_Answerを作成



プロパティ	値
名前	TextBlock_Answer
位置X	TextBlock_NumAの位置Xと同じ値
位置Y	120.0
サイズX	100.0
サイズY	150.0
Text	1.0
Color and Opacity	R:1.0 G:1.0 B:0.0 A:1.0
Font > Size	120

6. UMGの作成・表示（ウィジェットブループリント）

6.1 新規レベル（DispWidget）を作成・保存

6.2 画像ファイルのインポート

6.3 ウィジェットブループリント(WBP_MathGameOutline)を作成

6.4 ウィジェットブループリントエディタについて

6.5 ウィジェットの配置

6.5.1 Imageを使用してアウトライン背景を表示

6.5.2 Textを配置して計算式を作成

6.5.3 HorizonBoxを使用して、1~9のButtonを横並びに配置する

6.5.4 Imageを使用して結果判定の画像を配置

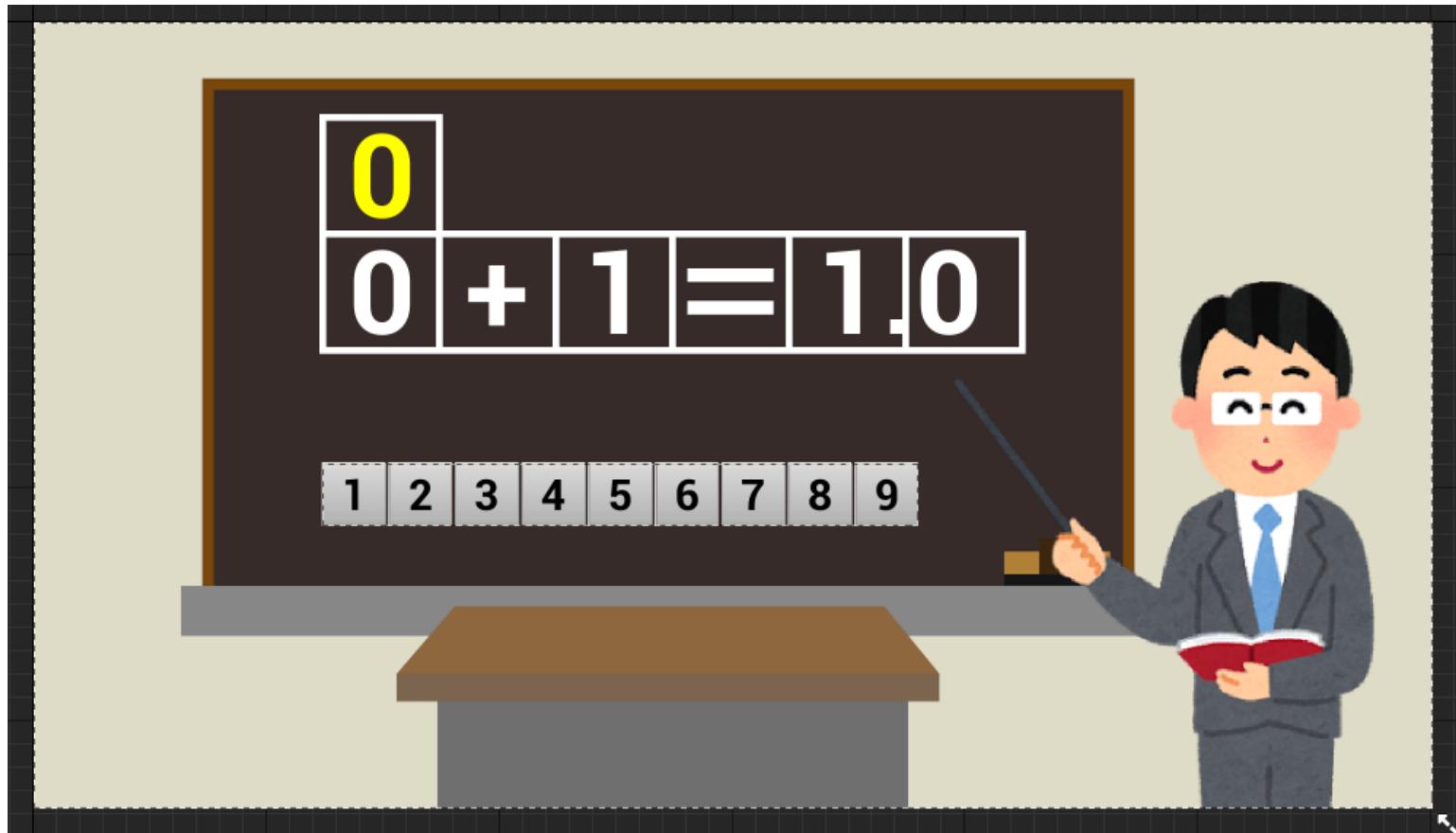
6.6 レベルブループリントの編集

6.6.1 ウィジェットを表示

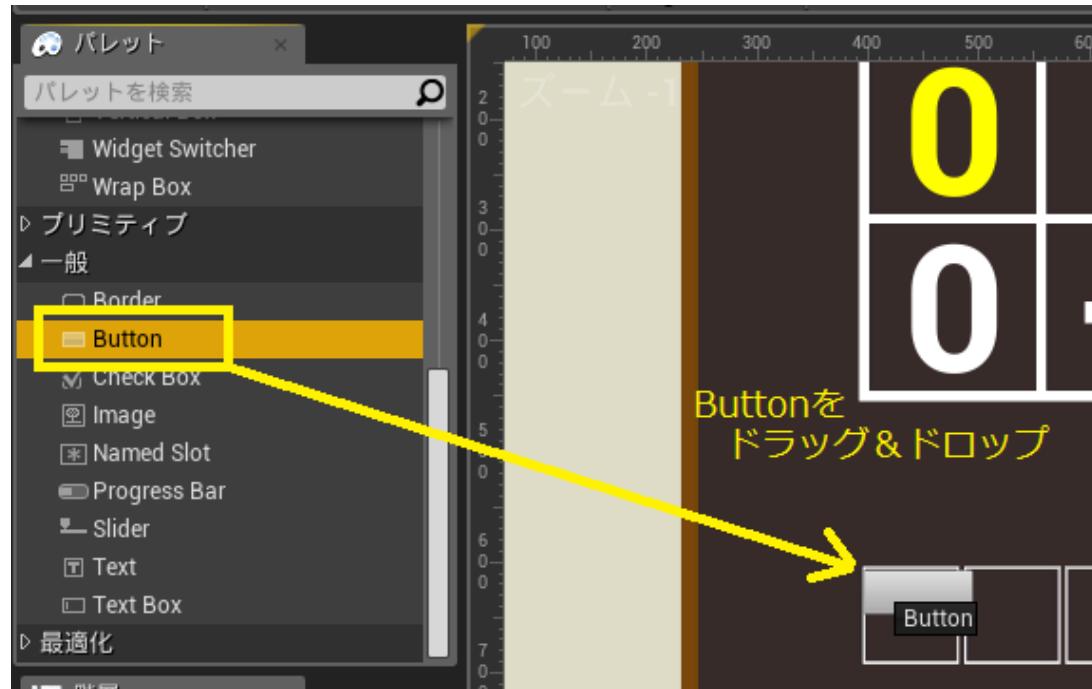
6.6.2 マウスカーソルを常に表示

6.6.3 ESCキーでゲームを終了

Buttonを1~9まで横並びに配置



Buttonをウィジェットデザイナーに追加



パレットタブ > 一般 > Buttonを
ウィジェットデザイナーにドラッグ&ドロップ

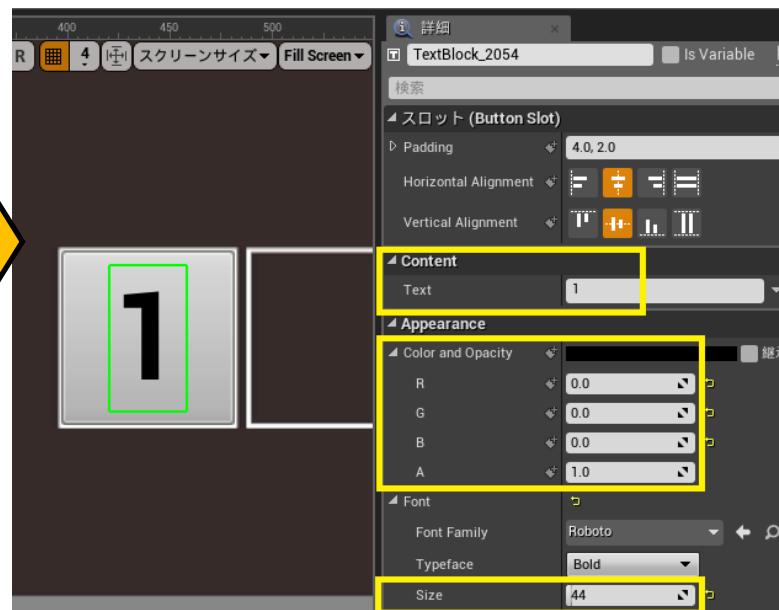
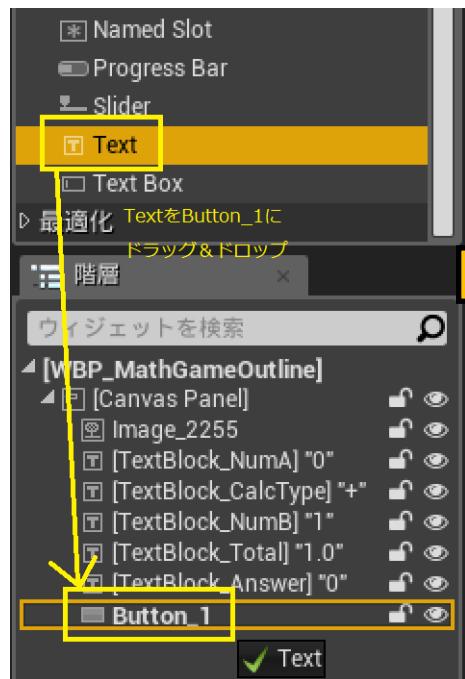
Buttonの位置とサイズを画像に合わせる



プロパティ	値
名前	Button_1
位置X	画像の□に合うように設定
位置Y	画像の□に合うように設定
サイズX	85.0
サイズY	85.0

ボタンの位置を画像に合わせる
ボタンの位置を合わせる際にスナップするサイズを調整する

TextをButton1に追加

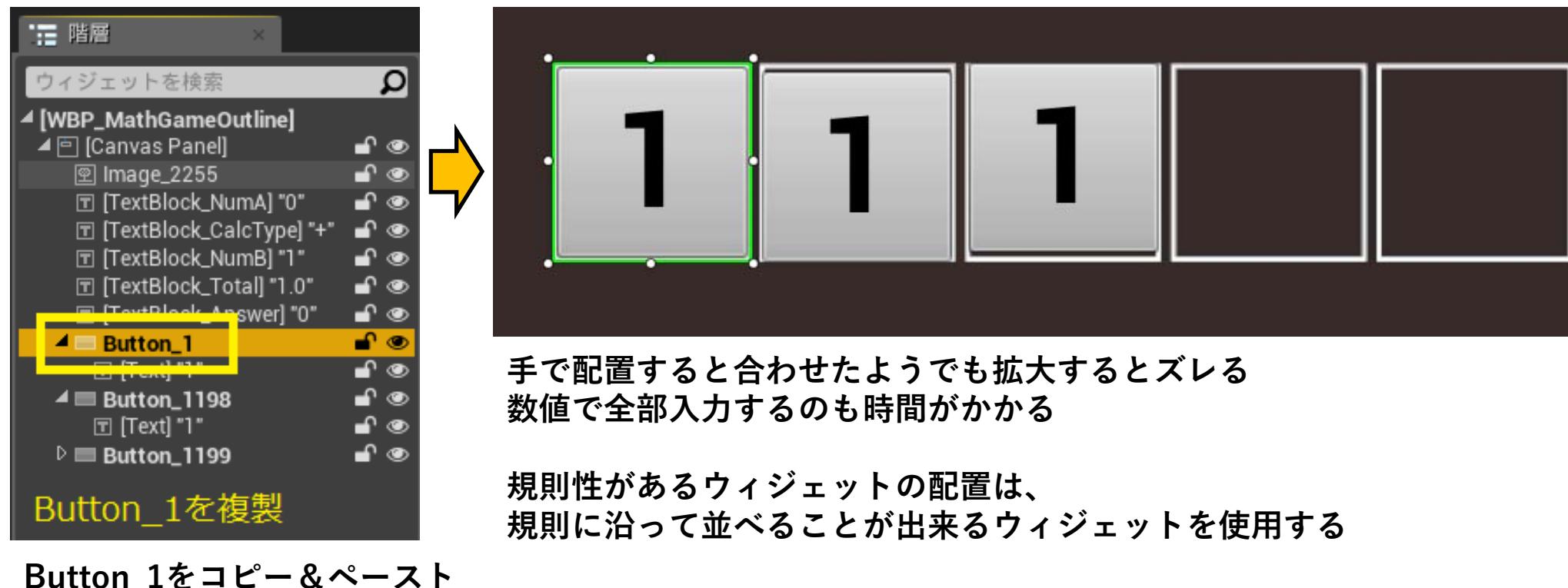


プロパティ	値
Text	1
Color and Opacity	R:0.0 G:0.0 B:0.0 A:1.0
Font > Size	44

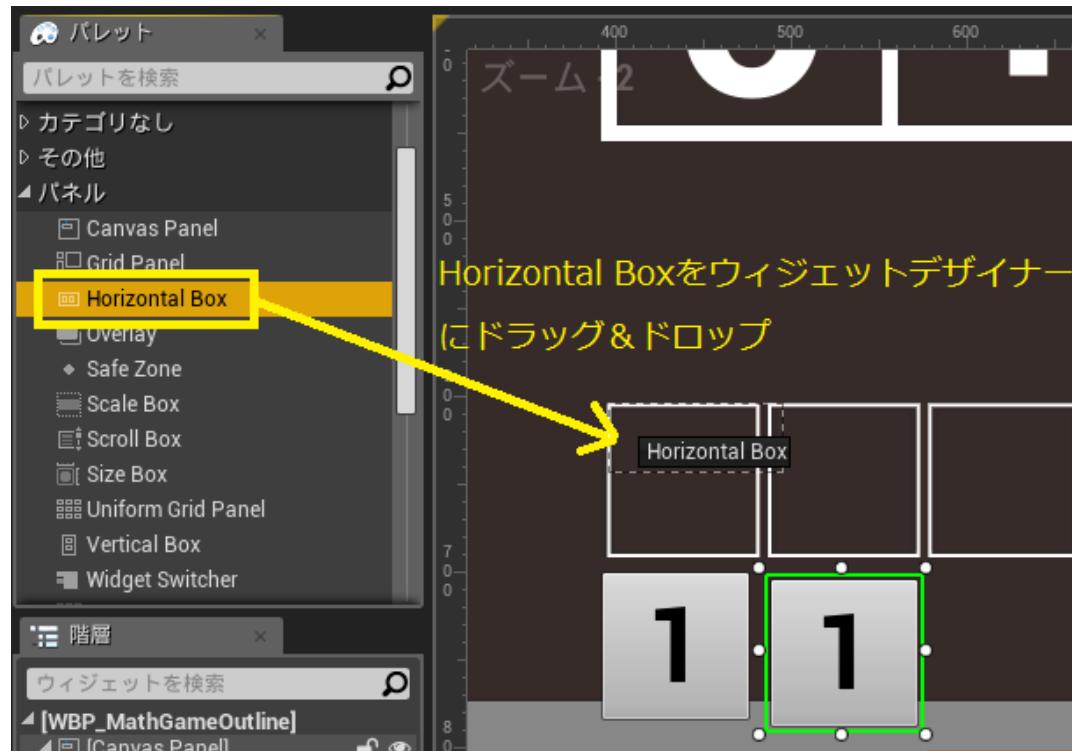
TextをButton_1に
ドラッグ&ドロップ

Textの詳細を設定

Button1を複製すれば、ButtonとTextのセットが複製できる
手で配置すると間隔や縦位置がズレてしまう

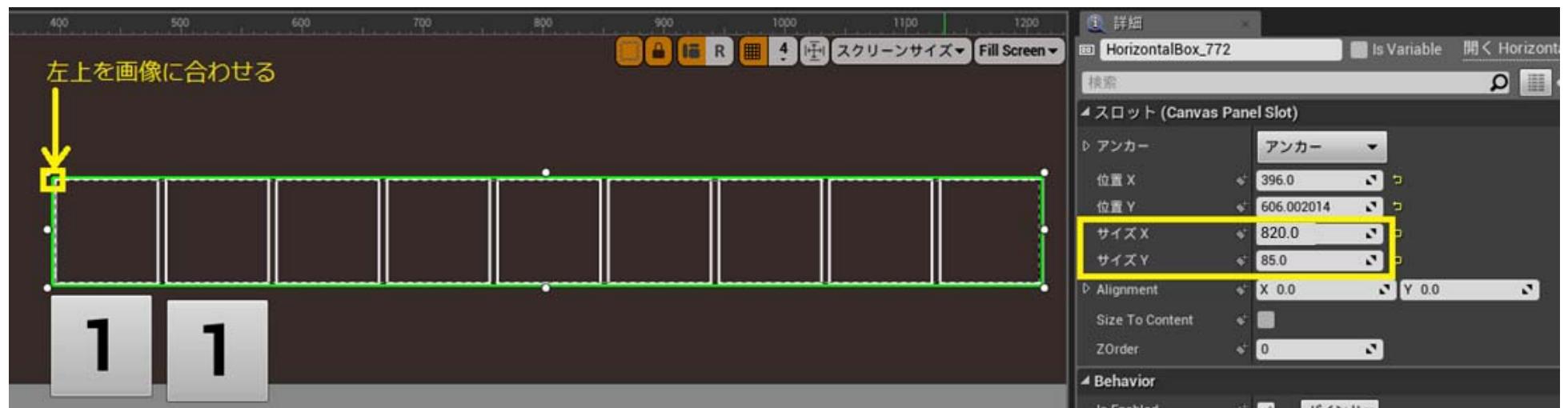


Horizontal Boxをウィジェットデザイナーにドラッグ&ドロップ



パネル > Horizontal Boxをウィジェットデザイナーに
ドラッグ&ドロップ

Horizontal Boxを画像に合わせる



サイズX,サイズYを設定してから
Horizontal Boxの左上を画像に合わせる

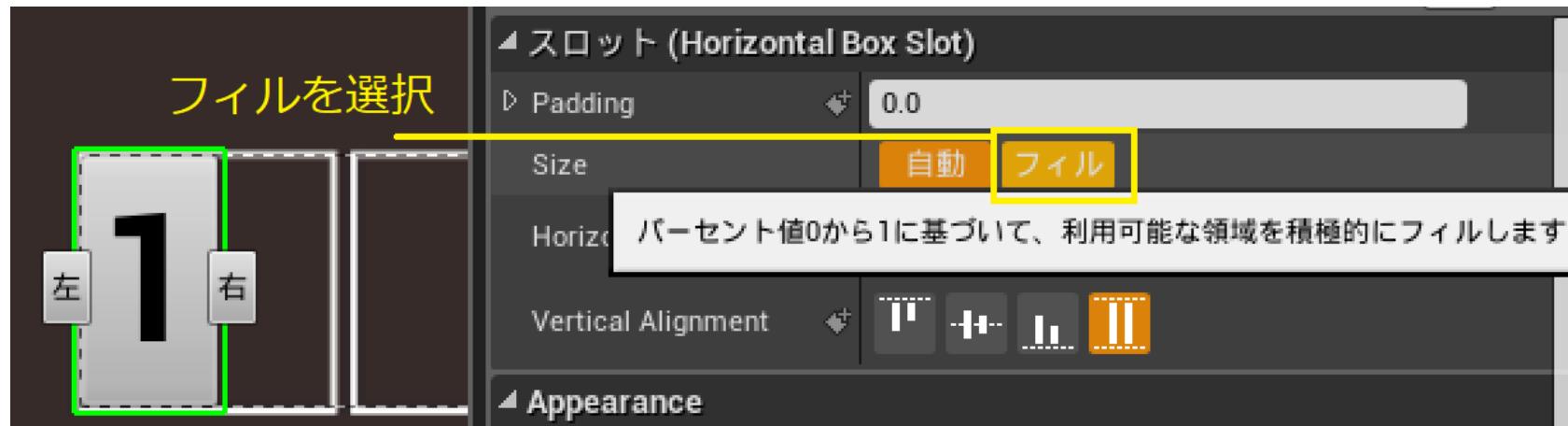
プロパティ	値
サイズX	820
サイズY	85

Button_1をHorizontal Boxにドラッグ＆ドロップ



Button_1をHorizontal Boxに
ドラッグ&ドロップ

Button_1のSizeをファイルに設定

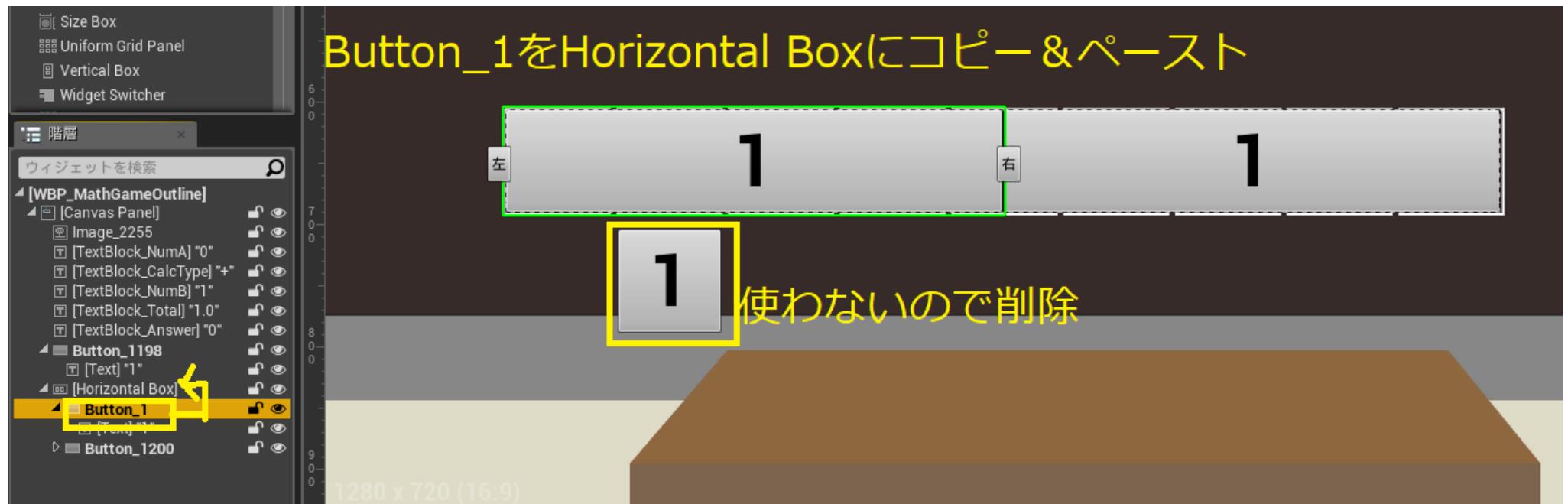


Button_1のSizeをファイルに設定



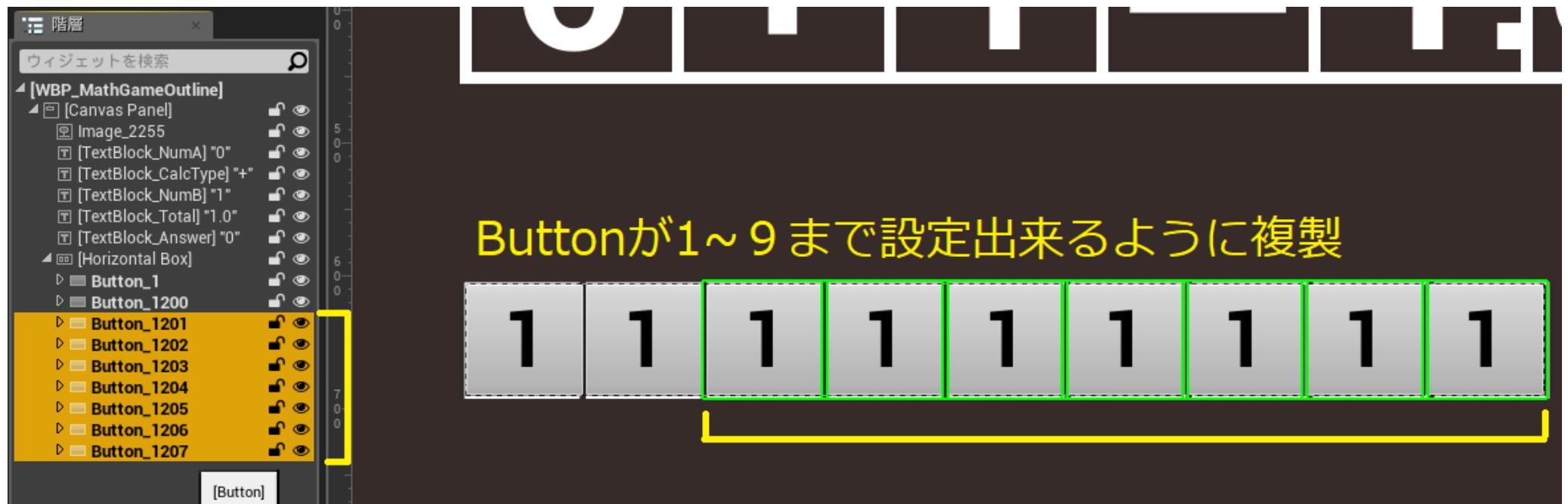
Button_1がHorizontal Box全体に広がる

Button_1をHorizontal Boxにコピー & ペースト



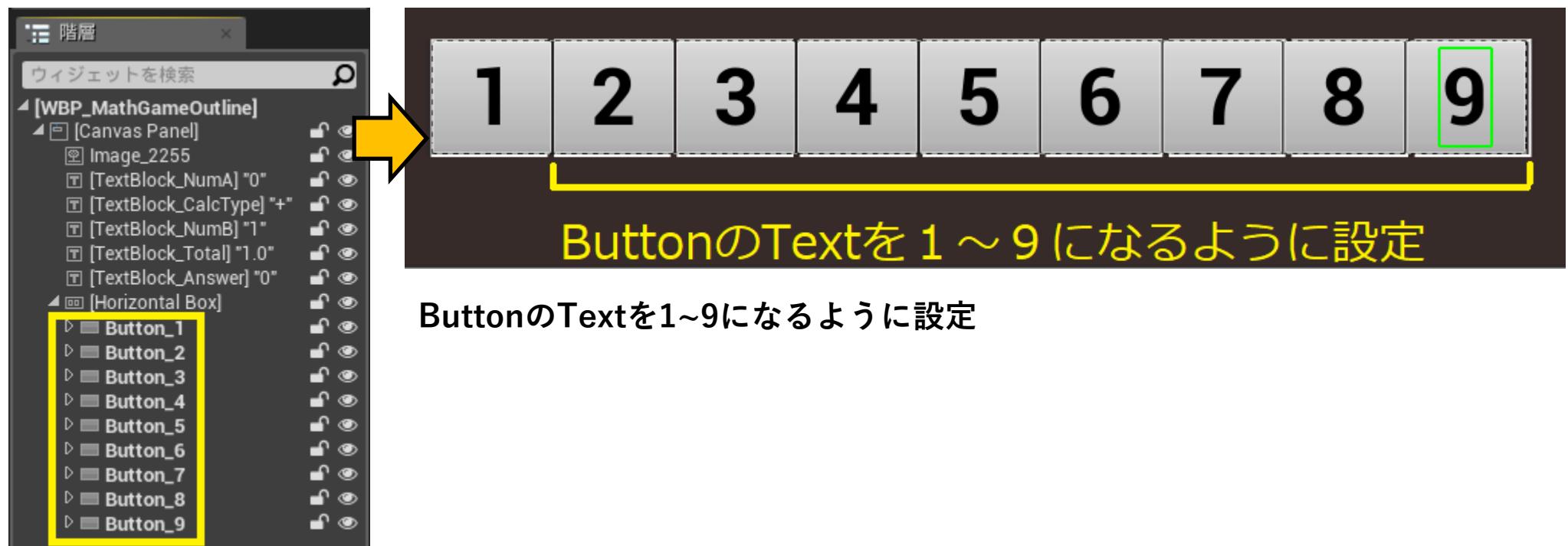
Button_1をHorizontal Boxにコピー & ペースト
Horizontal Boxにボタンが均等な大きさで配置される

Buttonが1~9まで設定できるように
Button_1を複製



Buttonを1~9まで設定できるようにHorizontal Boxにコピー & ペーストして複製

1~9になるように
Buttonの変数名,ButtonのTextを変更



Buttonの変数名を変更

ButtonのTextを1~9になるように設定

ButtonのTextを1~9になるように設定

Button間に余白がないので余白を設定

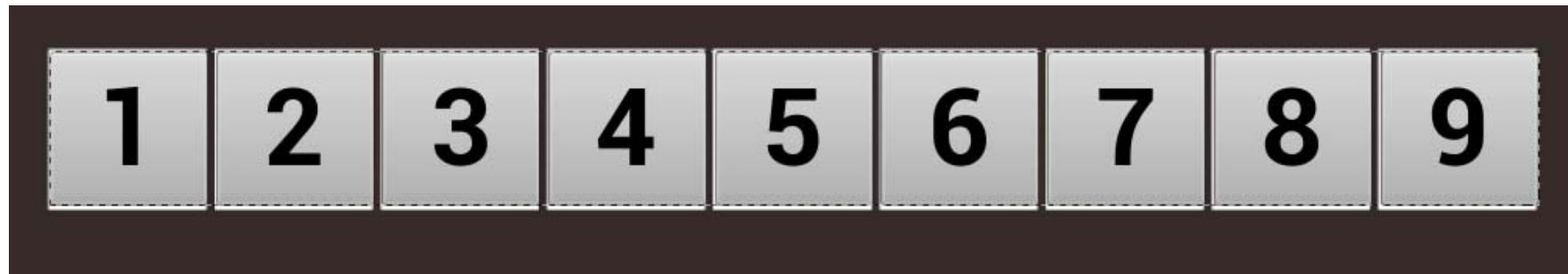


Horizontal Boxに配置した
Button間に余白がないので
調整する



プロパティ	値
Padding Left	2.0
Padding Right	2.0

Button_1～Button_9までPaddingを設定



Button_1

プロパティ	値
Padding Left	0.0
Padding Right	2.0

Button_2～Button_8

プロパティ	値
Padding Left	2.0
Padding Right	2.0

Button_9

プロパティ	値
Padding Left	2.0
Padding Right	0.0

6. UMGの作成・表示（ウィジェットブループリント）

6.1 新規レベル（DispWidget）を作成・保存

6.2 画像ファイルのインポート

6.3 ウィジェットブループリント(WBP_MathGameOutline)を作成

6.4 ウィジェットブループリントエディタについて

6.5 ウィジェットの配置

 6.5.1 Imageを使用してアウトライン背景を表示

 6.5.2 Textを配置して計算式を作成

 6.5.3 HorizonBoxを使用して、1~9のButtonを横並びに配置する

 6.5.4 Imageを使用して結果判定の画像を配置

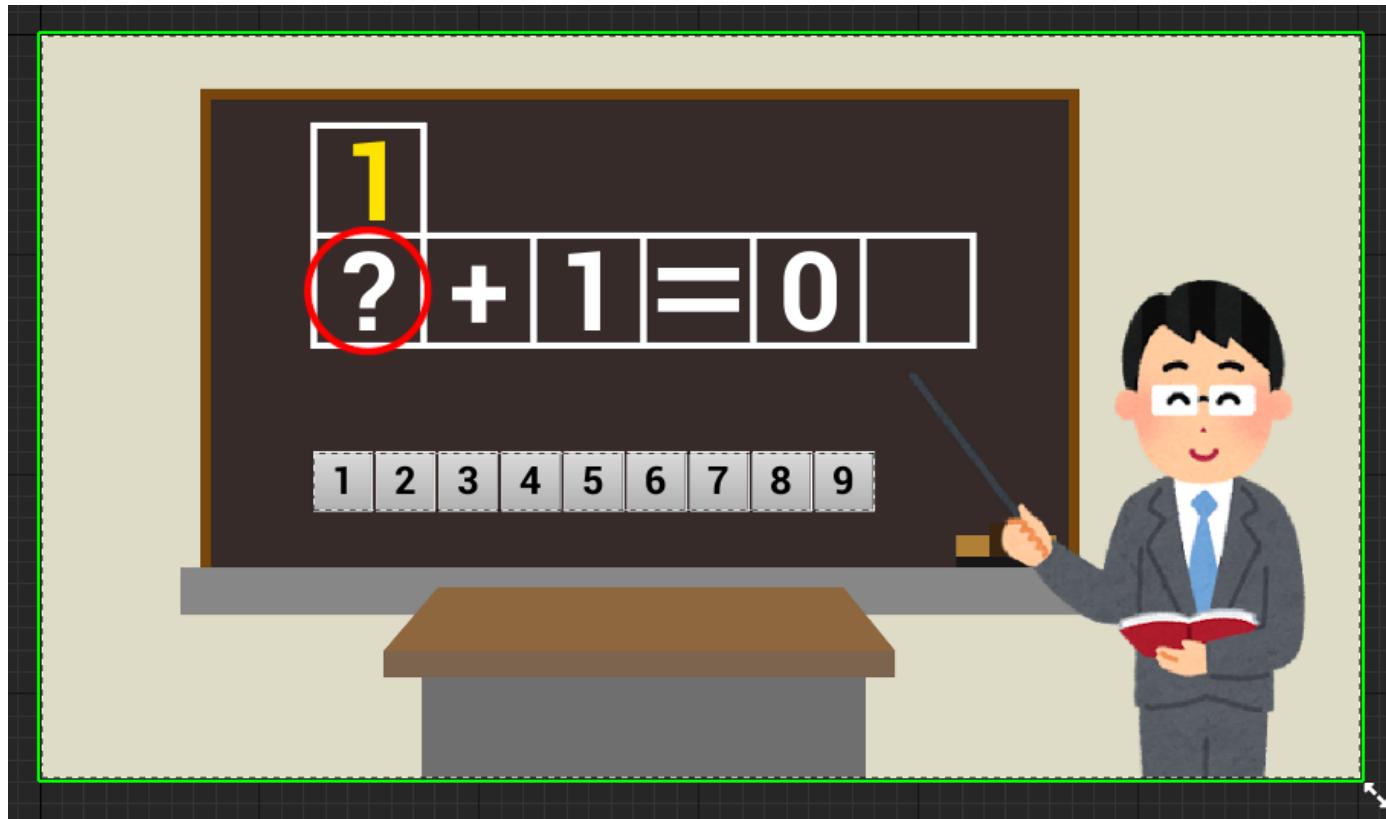
6.6 レベルブループリントの編集

 6.6.1 ウィジェットを表示

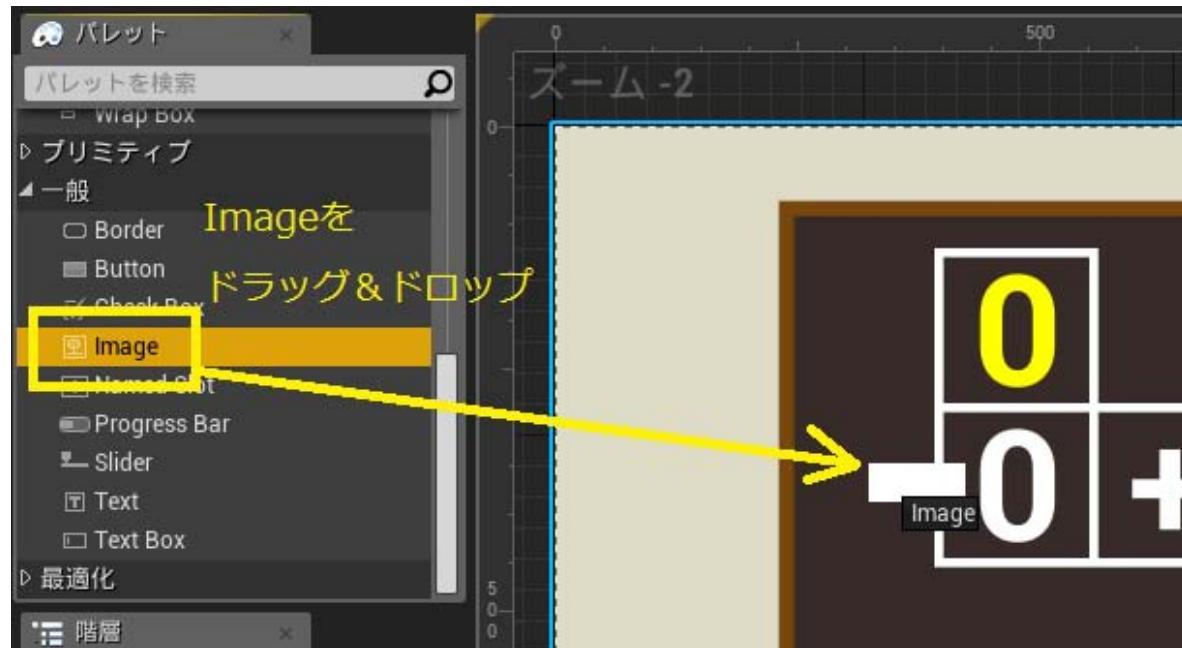
 6.6.2 マウスカーソルを常に表示

 6.6.3 ESCキーでゲームを終了

結果判定の画像を配置

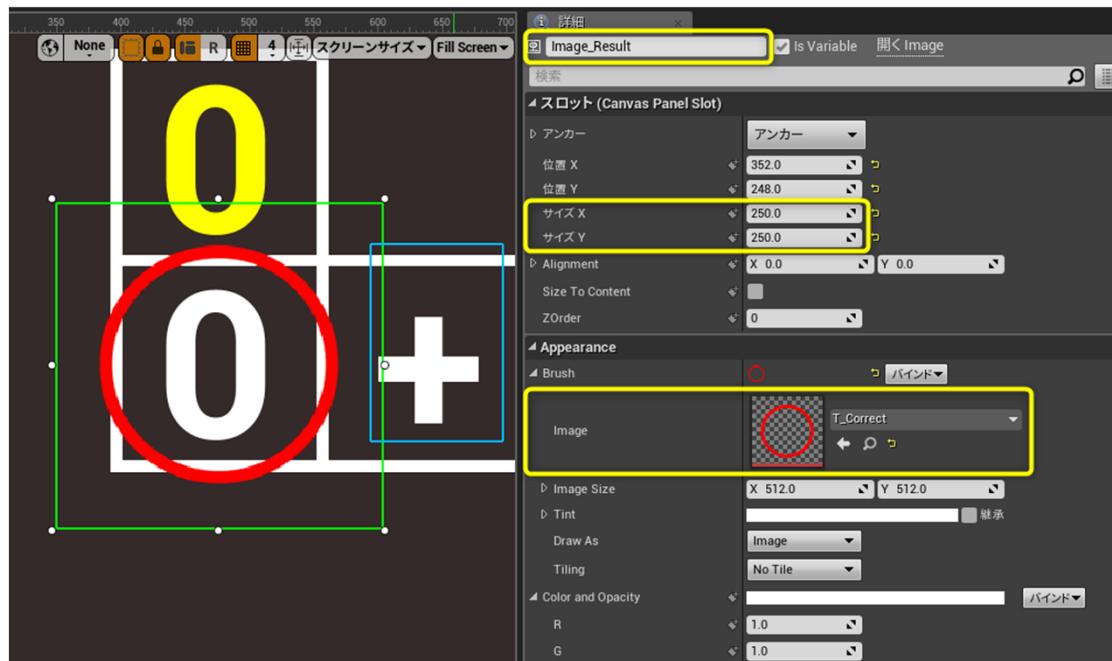


Imageをウィジェットデザイナーに追加



パレットタブ > 一般 > Imageを
ウィジェットデザイナーにドラッグ&ドロップ

結果判定のImageの詳細を設定



プロパティ	値
名前	Image.Result
サイズX	250.0
サイズY	250.0
Brush Image	T_Correct

6. UMGの作成・表示（ウィジェットブループリント）

6.1 新規レベル（DispWidget）を作成・保存

6.2 画像ファイルのインポート

6.3 ウィジェットブループリント(WBP_MathGameOutline)を作成

6.4 ウィジェットブループリントエディタについて

6.5 ウィジェットの配置

 6.5.1 Imageを使用してアウトライン背景を表示

 6.5.2 Textを配置して計算式を作成

 6.5.3 HorizonBoxを使用して、1~9のButtonを横並びに配置する

 6.5.4 Imageを使用して結果判定の画像を配置

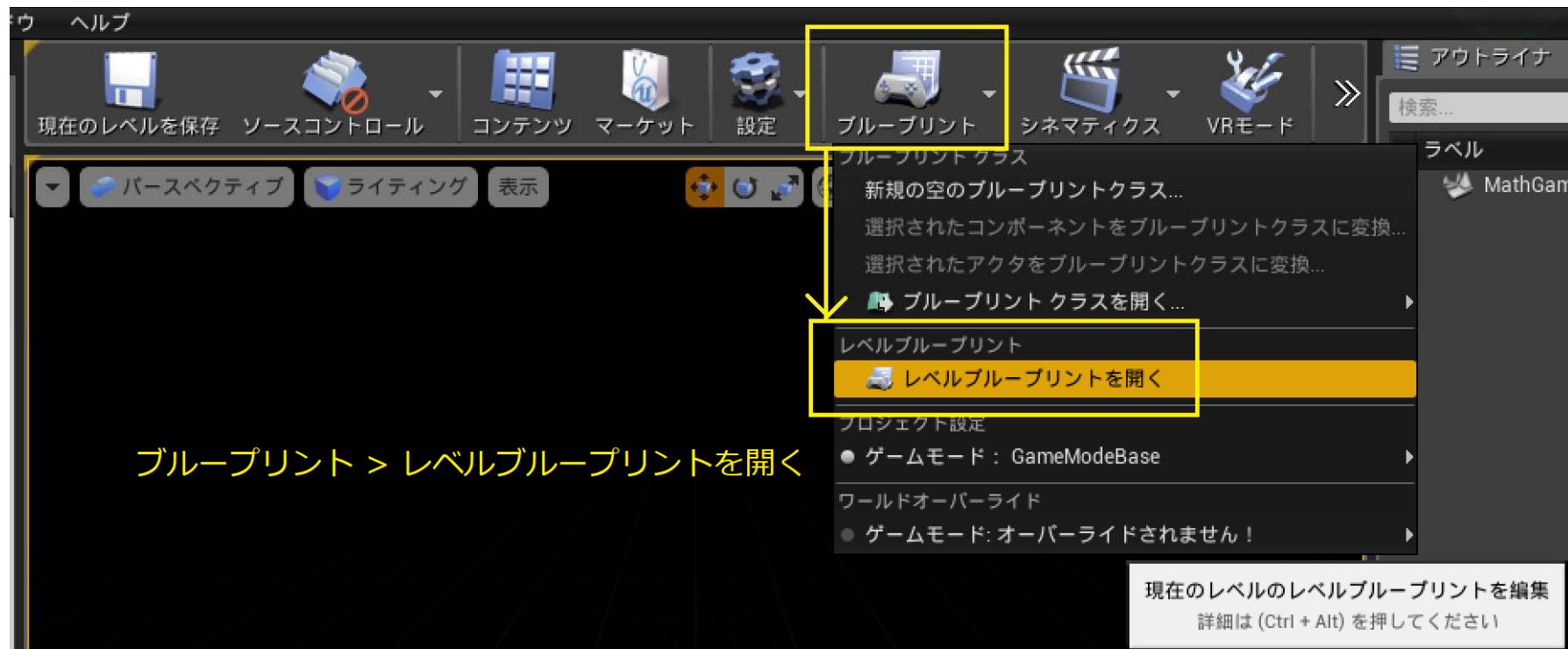
6.6 レベルブループリントの編集

 6.6.1 ウィジェットを表示

 6.6.2 マウスカーソルを常に表示

 6.6.3 ESCキーでゲームを終了

レベルブループリントを開く



ブループリント > ブループリントを開く

6. UMGの作成・表示（ウィジェットブループリント）

6.1 新規レベル（DispWidget）を作成・保存

6.2 画像ファイルのインポート

6.3 ウィジェットブループリント(WBP_MathGameOutline)を作成

6.4 ウィジェットブループリントエディタについて

6.5 ウィジェットの配置

6.5.1 Imageを使用してアウトライン背景を表示

6.5.2 Textを配置して計算式を作成

6.5.3 HorizonBoxを使用して、1~9のButtonを横並びに配置する

6.5.4 Imageを使用して結果判定の画像を配置

6.6 レベルブループリントの編集

6.6.1 ウィジェットを表示

6.6.2 マウスカーソルを常に表示

6.6.3 ESCキーでゲームを終了

ウィジェット作成ノードを追加



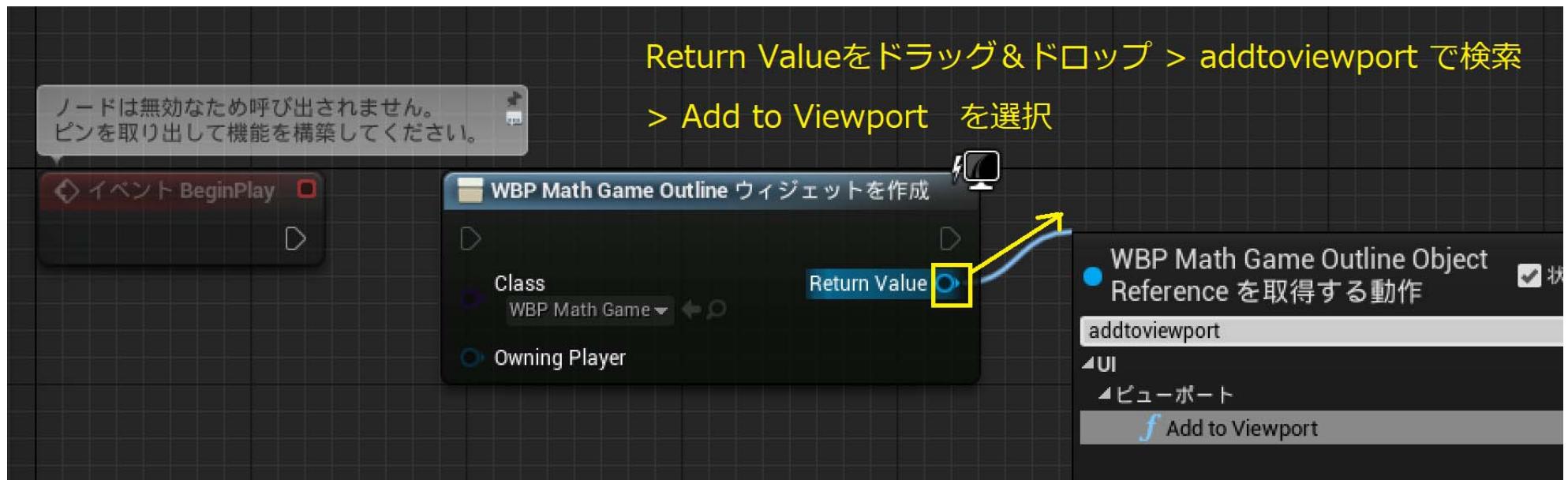
右クリック > `createwidget` で検索 > ウィジェットを作成 を選択

Classのリストから WBP_MathGameOutlineを選択



ウィジェット作成のClassのリストからWBP_MathGameOutlineを選択

ウィジェット作成ノードのReturn Value からAdd to Viewportを追加

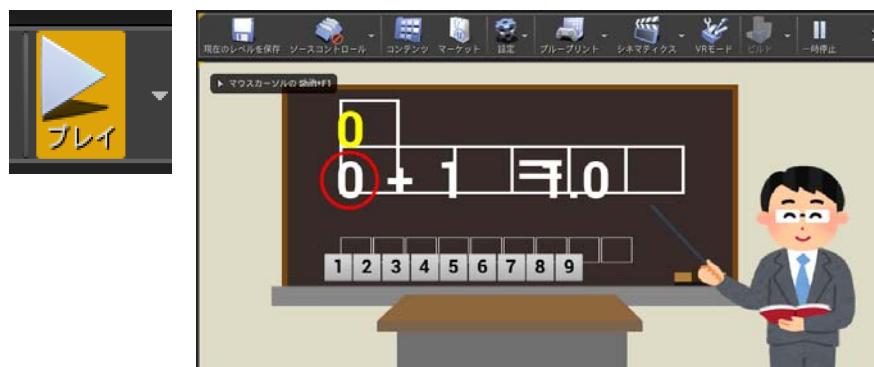


ウィジェットを作成のReturn Valueをドラッグ&ドロップ
> addtoviewport で検索
> [Add to Viewport]を選択

BeginPlayでウィジェットの表示処理を実行



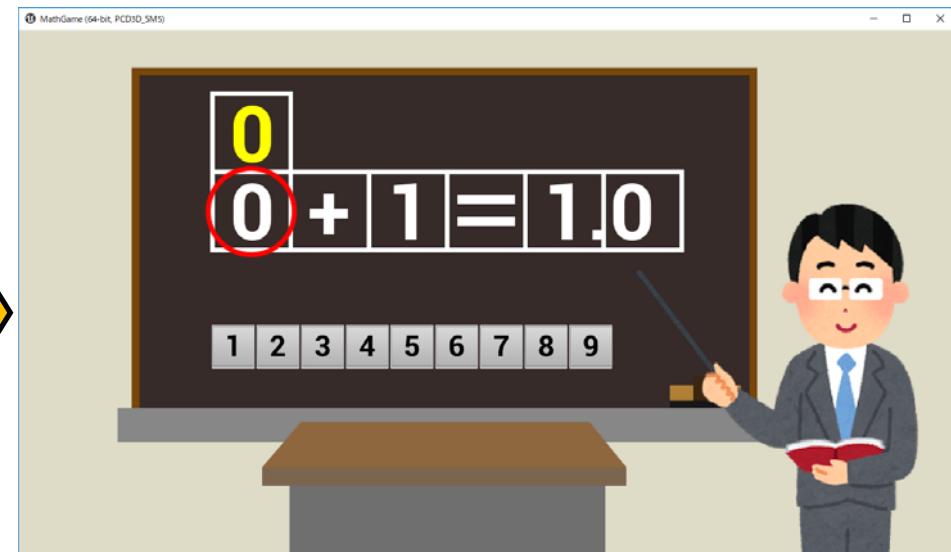
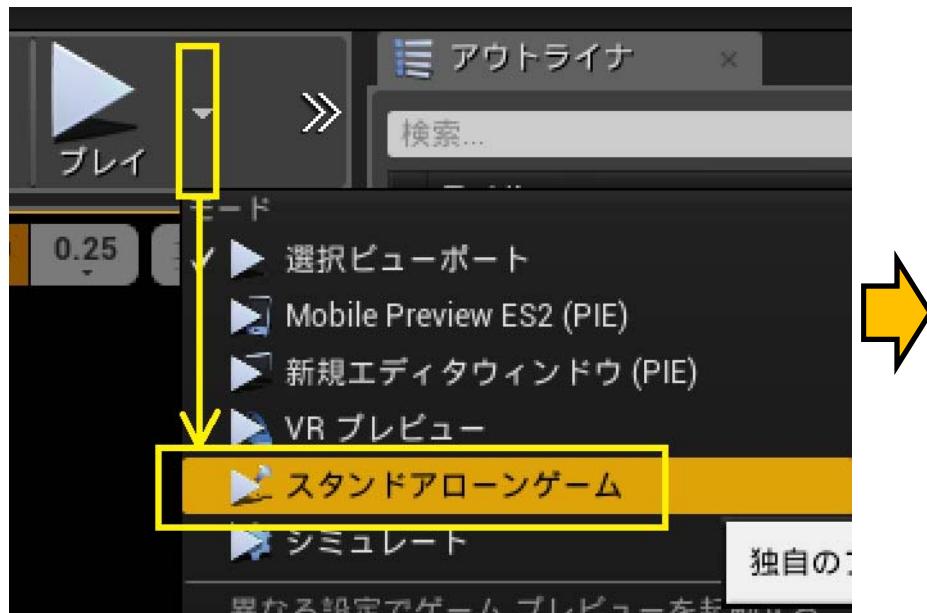
BeginPlayでウィジェットの表示処理を実行



ウィジェットが表示されるが、アスペクト比
が合っていないためレイアウトが崩れる

Shift + F1でマウスカーソルを表示
ボタンがクリックできる

スタンドアローンゲームで実行



プレイ右の▼クリック
> [スタンドアローンゲーム]を選択

レイアウトが崩れなくなる
ESCキーで終了、Shift + F1でカーソル表示ができなくなる
終了したい場合は、Alt + F4で終了

6. UMGの作成・表示（ウィジェットブループリント）

6.1 新規レベル（DispWidget）を作成・保存

6.2 画像ファイルのインポート

6.3 ウィジェットブループリント(WBP_MathGameOutline)を作成

6.4 ウィジェットブループリントエディタについて

6.5 ウィジェットの配置

6.5.1 Imageを使用してアウトライン背景を表示

6.5.2 Textを配置して計算式を作成

6.5.3 HorizonBoxを使用して、1~9のButtonを横並びに配置する

6.5.4 Imageを使用して結果判定の画像を配置

6.6 レベルブループリントの編集

6.6.1 ウィジェットを表示

6.6.2 マウスカーソルを常に表示

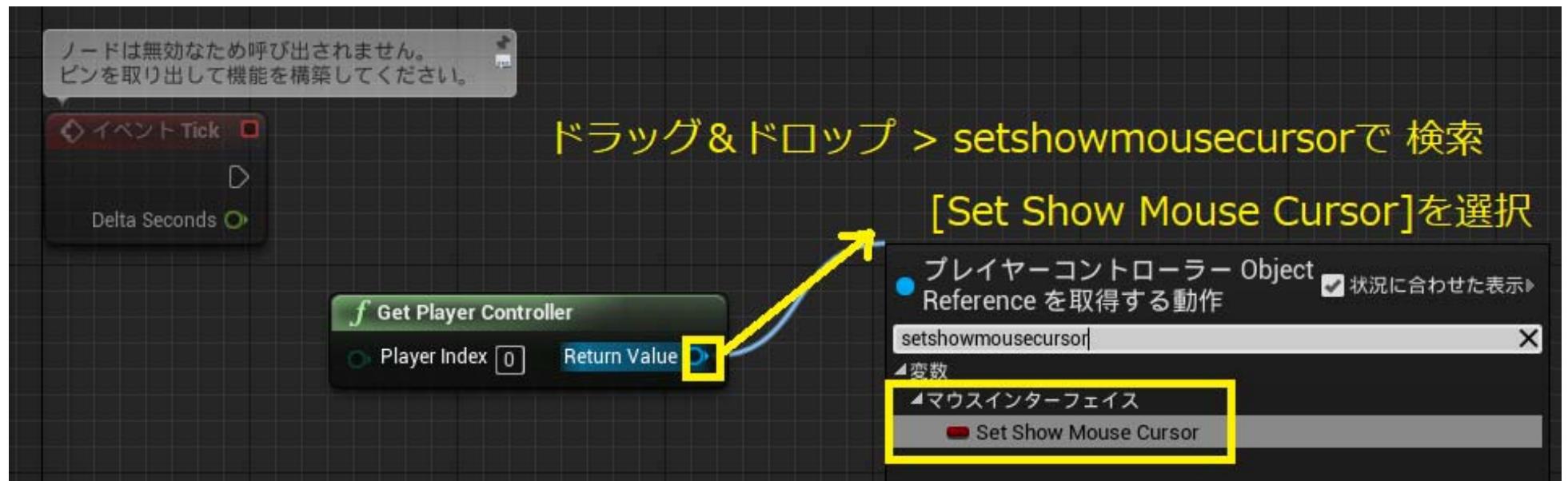
6.6.3 ESCキーでゲームを終了

Get Player Controllerを追加



右クリック > getplayercontroller で検索
> [Get Player Controller]を選択

Set Show Mouse Cursorを追加

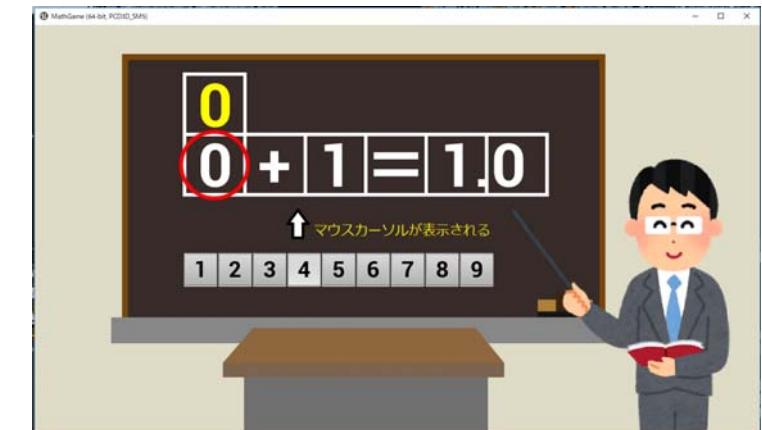


Get Player ControllerのReturn Valueを ドラッグ & ドロップ
> setshowmousecursor で検索
> [Set Show Mouse Cursor]を選択

イベントTickから マウスカーソル表示処理を呼び出す



Show Mouse Cursorにチェックを付ける
イベントTickからマウスカーソル表示処理を呼び出す



プレイして確認
マウスカーソルが表示され続ける

6. UMGの作成・表示（ウィジェットブループリント）

6.1 新規レベル（DispWidget）を作成・保存

6.2 画像ファイルのインポート

6.3 ウィジェットブループリント(WBP_MathGameOutline)を作成

6.4 ウィジェットブループリントエディタについて

6.5 ウィジェットの配置

 6.5.1 Imageを使用してアウトライン背景を表示

 6.5.2 Textを配置して計算式を作成

 6.5.3 HorizonBoxを使用して、1~9のButtonを横並びに配置する

 6.5.4 Imageを使用して結果判定の画像を配置

6.6 レベルブループリントの編集

 6.6.1 ウィジェットを表示

 6.6.2 マウスカーソルを常に表示

 6.6.3 ESCキーでゲームを終了

キーボードイベント ESCとQuit Gameを追加

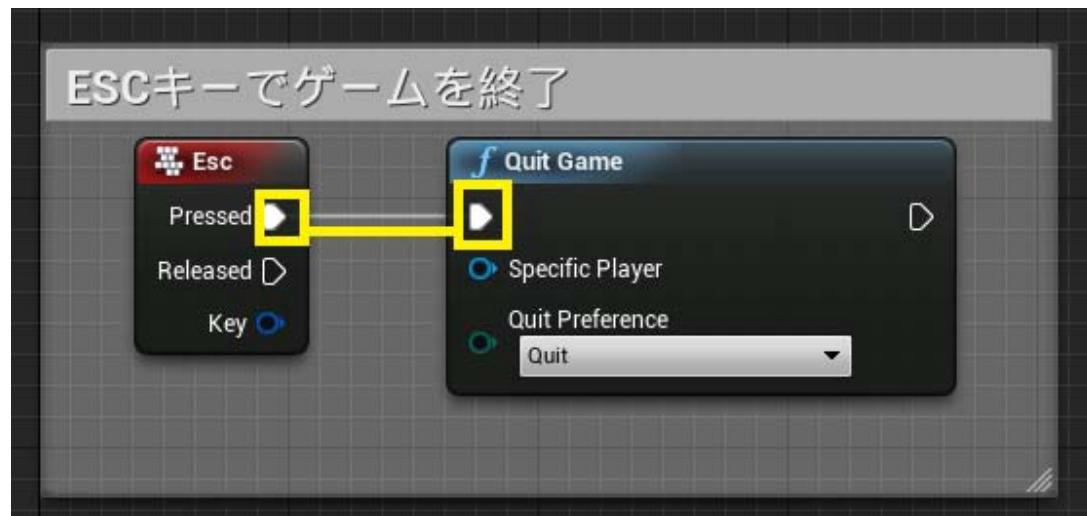


右クリック > ESC で検索
> キーボードイベント [ESC]を選択



右クリック > quit game で検索
> [Quit Game]を選択

ESCのPressedからQuit Gameを呼びだす



ESCキーからQuit Gameを呼び出す



プレイして確認
選択ビューポートでプレイと同様に、
ESCキーでゲーム終了

7. UMGからゲームを実行できるようにする

7. UMGからゲームを実行できるようにする

7.1 新規レベル（MathGame）を作成・保存

7.2 WBP_MathGameOutlineを複製して、WBP_MathGameを作成

7.3 背景をアウトラインがない画像に変更

7.4 WBP_MathGameOutlineからWBP_MathGameを表示するようにレベルブループリントを修正

7.5 ウィジェットブループリントの基本機能を実装

 7.5.1 ButtonのClickイベント

 7.5.2 TextのTextプロパティの値を変更

 7.5.3 Imageの画像切り替え

 7.5.4 Imageの表示/非表示

7.6 BP_RandomCalcGameの処理を参考にWBP_MathGameを実装

 7.6.1 BP_RandomCalcGameから変更するポイント

 7.6.2 BP_RandomCalcGameから変数、RandomCalculationを移植

 7.6.3 ボタンのクリックイベントをTextBlock_NumAのTextに設定

 7.6.4 計算とボタンの入力値を判定して結果画像と答えを表示

 7.6.5 計算と入力判定のループ化

7. UMGからゲームを実行できるようにする

7.1 新規レベル（MathGame）を作成・保存

7.2 WBP_MathGameOutlineを複製して、WBP_MathGameを作成

7.3 背景をアウトラインがない画像に変更

7.4 WBP_MathGameOutlineからWBP_MathGameを表示するようにレベルブループリントを修正

7.5 ウィジェットブループリントの基本機能を実装

 7.5.1 ButtonのClickイベント

 7.5.2 TextのTextプロパティの値を変更

 7.5.3 Imageの画像切り替え

 7.5.4 Imageの表示/非表示

7.6 BP_RandomCalcGameの処理を参考にWBP_MathGameを実装

 7.6.1 BP_RandomCalcGameから変更するポイント

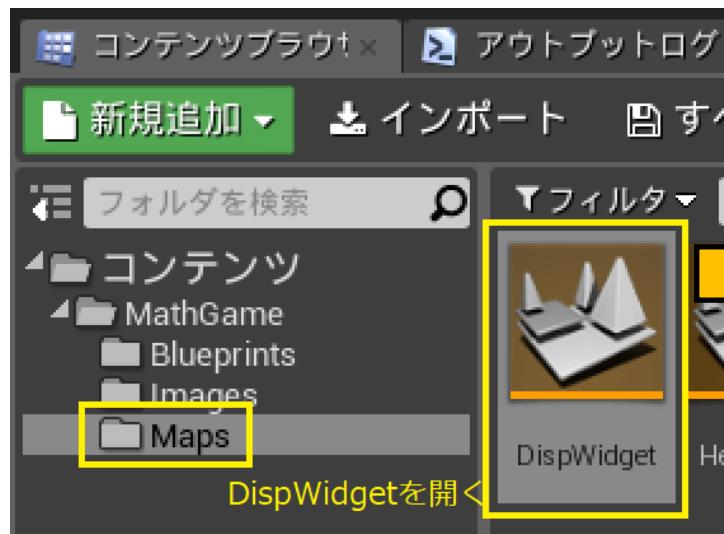
 7.6.2 BP_RandomCalcGameから変数、RandomCalculationを移植

 7.6.3 ボタンのクリックイベントをTextBlock_NumAのTextに設定

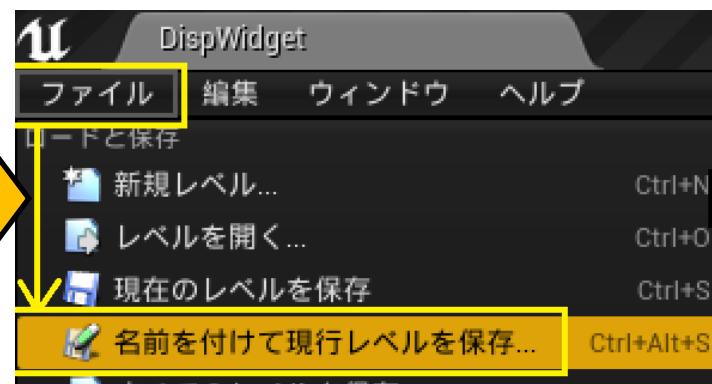
 7.6.4 計算とボタンの入力値を判定して結果画像と答えを表示

 7.6.5 計算と入力判定のループ化

DispWidgetを名前を付けて保存（MathGame）



DispWidgetを開く



ファイル
> 名前を付けて現行レベルを保存



Mapsを選択
> 名前を[MathGame]に設定
> 保存

7. UMGからゲームを実行できるようにする

7.1 新規レベル（MathGame）を作成・保存

7.2 WBP_MathGameOutlineを複製して、WBP_MathGameを作成

7.3 背景をアウトラインがない画像に変更

7.4 WBP_MathGameOutlineからWBP_MathGameを表示するようにレベルブループリントを修正

7.5 ウィジェットブループリントの基本機能を実装

 7.5.1 ButtonのClickイベント

 7.5.2 TextのTextプロパティの値を変更

 7.5.3 Imageの画像切り替え

 7.5.4 Imageの表示/非表示

7.6 BP_RandomCalcGameの処理を参考にWBP_MathGameを実装

 7.6.1 BP_RandomCalcGameから変更するポイント

 7.6.2 BP_RandomCalcGameから変数、RandomCalculationを移植

 7.6.3 ボタンのクリックイベントをTextBlock_NumAのTextに設定

 7.6.4 計算とボタンの入力値を判定して結果画像と答えを表示

 7.6.5 計算と入力判定のループ化

WBP_MathGameOutlineを複製して、WBP_MathGameを作成



WBP_MathGameOutlineを複製してWBP_MathGameを作成

7. UMGからゲームを実行できるようにする

7.1 新規レベル（MathGame）を作成・保存

7.2 WBP_MathGameOutlineを複製して、WBP_MathGameを作成

7.3 背景をアウトラインがない画像に変更

7.4 WBP_MathGameOutlineからWBP_MathGameを表示するようにレベルブループリントを修正

7.5 ウィジェットブループリントの基本機能を実装

 7.5.1 ButtonのClickイベント

 7.5.2 TextのTextプロパティの値を変更

 7.5.3 Imageの画像切り替え

 7.5.4 Imageの表示/非表示

7.6 BP_RandomCalcGameの処理を参考にWBP_MathGameを実装

 7.6.1 BP_RandomCalcGameから変更するポイント

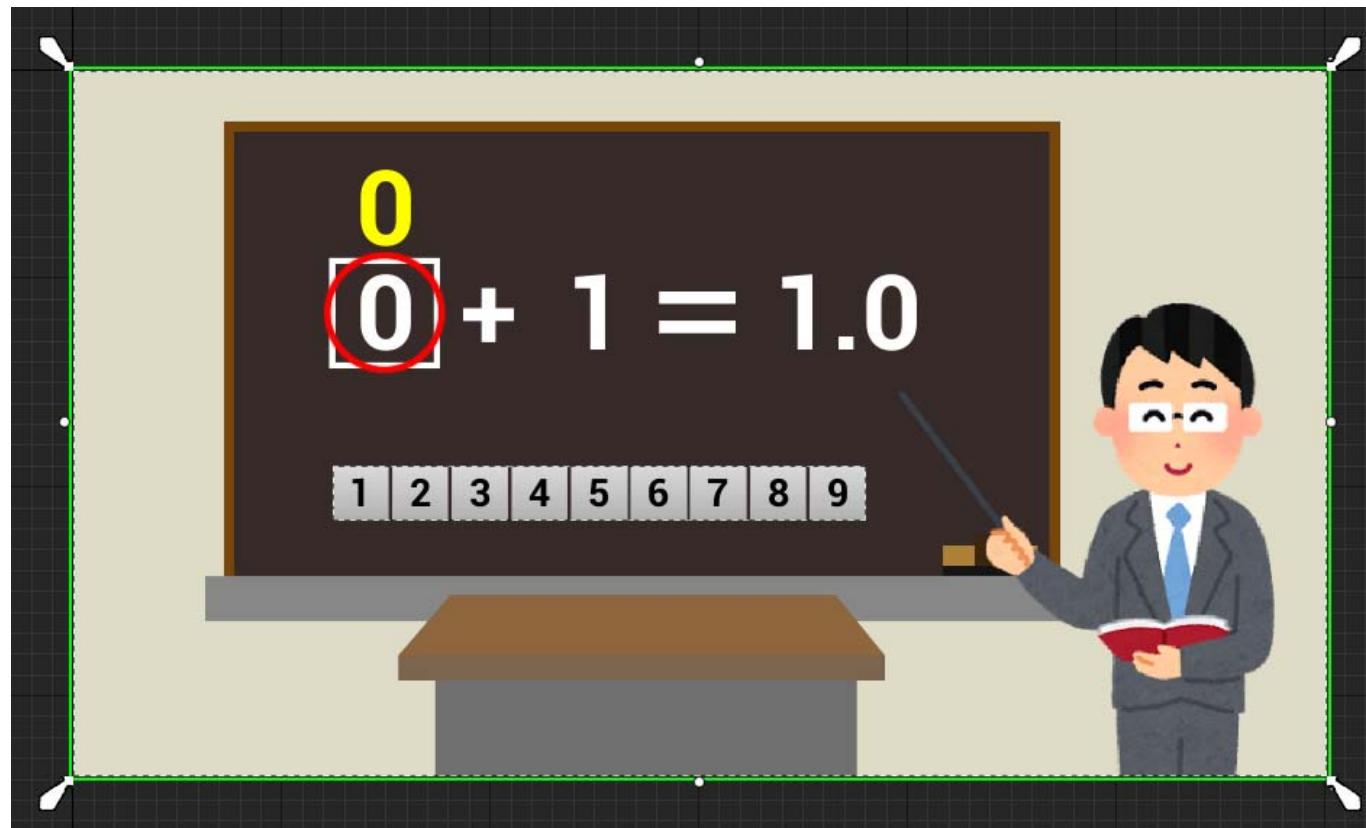
 7.6.2 BP_RandomCalcGameから変数、RandomCalculationを移植

 7.6.3 ボタンのクリックイベントをTextBlock_NumAのTextに設定

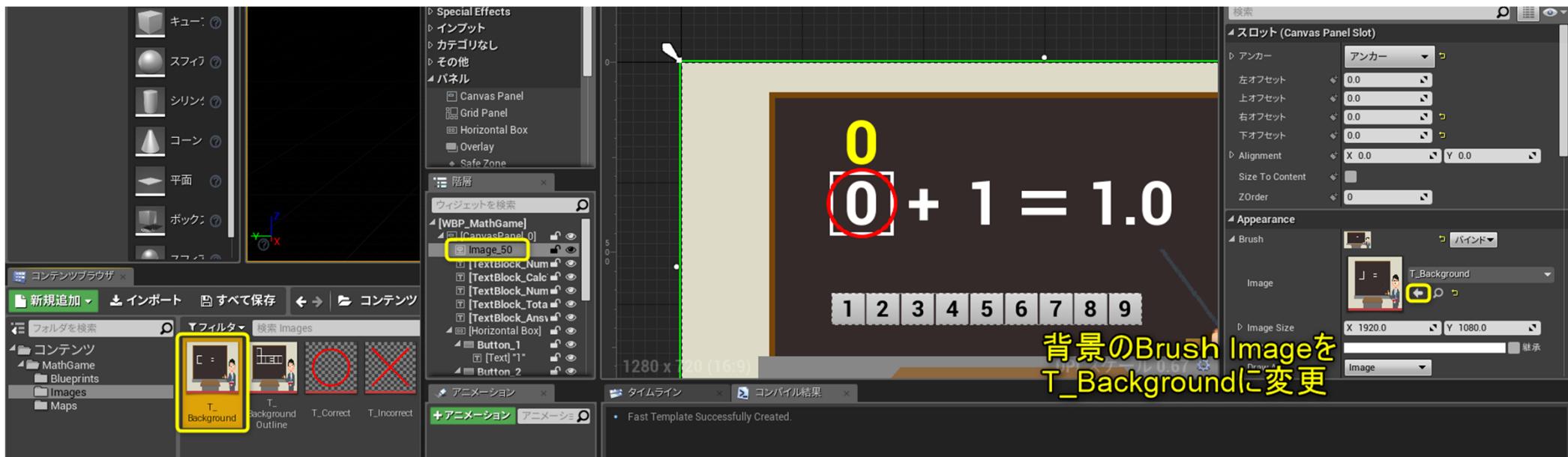
 7.6.4 計算とボタンの入力値を判定して結果画像と答えを表示

 7.6.5 計算と入力判定のループ化

背景をアウトラインがない画像に変更



背景のBrush ImageをT_Backgroudに変更



背景のBrush ImageをT_Backgroundに変更

7. UMGからゲームを実行できるようにする

7.1 新規レベル（MathGame）を作成・保存

7.2 WBP_MathGameOutlineを複製して、WBP_MathGameを作成

7.3 背景をアウトラインがない画像に変更

7.4 WBP_MathGameOutlineからWBP_MathGameを表示するようにレベルブループリントを修正

7.5 ウィジェットブループリントの基本機能を実装

 7.5.1 ButtonのClickイベント

 7.5.2 TextのTextプロパティの値を変更

 7.5.3 Imageの画像切り替え

 7.5.4 Imageの表示/非表示

7.6 BP_RandomCalcGameの処理を参考にWBP_MathGameを実装

 7.6.1 BP_RandomCalcGameから変更するポイント

 7.6.2 BP_RandomCalcGameから変数、RandomCalculationを移植

 7.6.3 ボタンのクリックイベントをTextBlock_NumAのTextに設定

 7.6.4 計算とボタンの入力値を判定して結果画像と答えを表示

 7.6.5 計算と入力判定のループ化

表示するウィジェットClassを WBP_MathGameに変更



ブループリント > レベルブループリントを開く



ウィジェットを作成のClassを
WBP_MathGameに変更

7. UMGからゲームを実行できるようにする

7.1 新規レベル（MathGame）を作成・保存

7.2 WBP_MathGameOutlineを複製して、WBP_MathGameを作成

7.3 背景をアウトラインがない画像に変更

7.4 WBP_MathGameOutlineからWBP_MathGameを表示するようにレベルブループリントを修正

7.5 ウィジェットブループリントの基本機能を実装

 7.5.1 ButtonのClickイベント

 7.5.2 TextのTextプロパティの値を変更

 7.5.3 Imageの画像切り替え

 7.5.4 Imageの表示/非表示

7.6 BP_RandomCalcGameの処理を参考にWBP_MathGameを実装

 7.6.1 BP_RandomCalcGameから変更するポイント

 7.6.2 BP_RandomCalcGameから変数、RandomCalculationを移植

 7.6.3 ボタンのクリックイベントをTextBlock_NumAのTextに設定

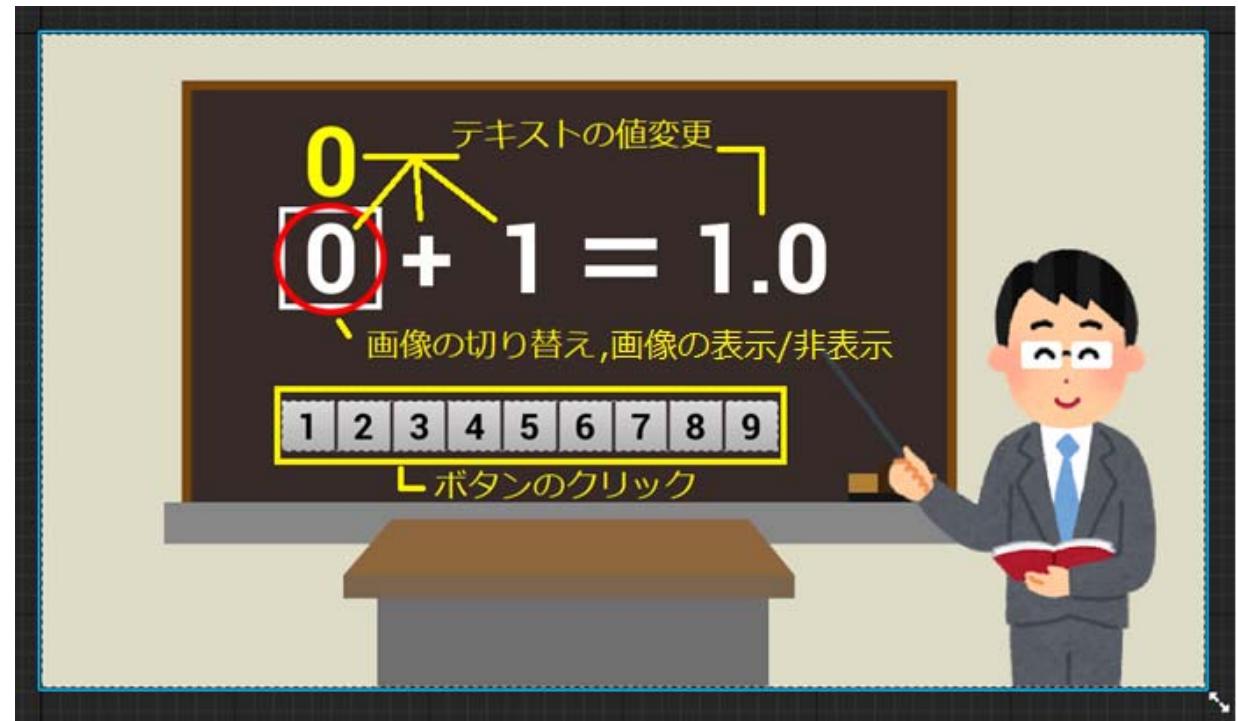
 7.6.4 計算とボタンの入力値を判定して結果画像と答えを表示

 7.6.5 計算と入力判定のループ化

ウィジェットブループリントの基本機能を実装

今回のゲームに必要な
基本機能を実装

- ・ボタンのクリック
- ・テキストの値変更
- ・画像の切り替え
- ・画像の表示/非表示



ウィジェットブループリントの処理



ボタンのクリックイベント

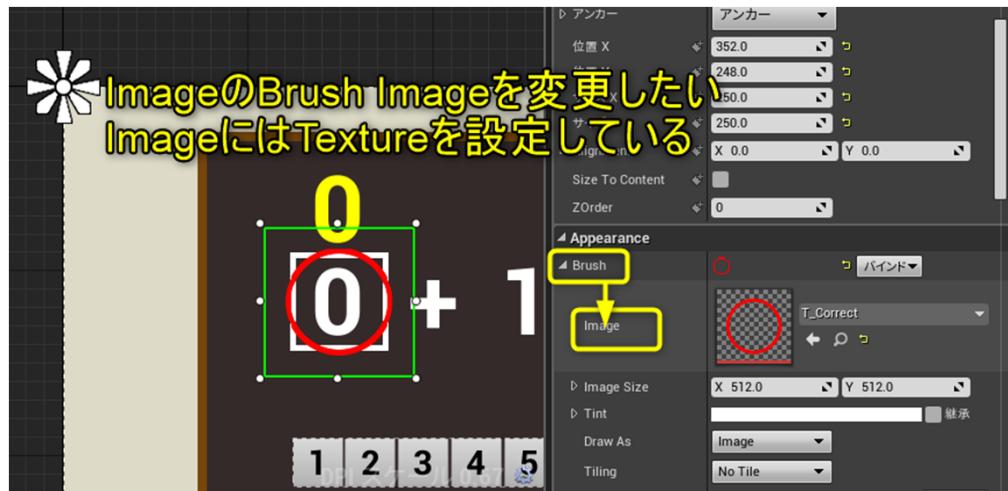
TextのTextプロパティを変更

Imageの画像(Texture)を変更

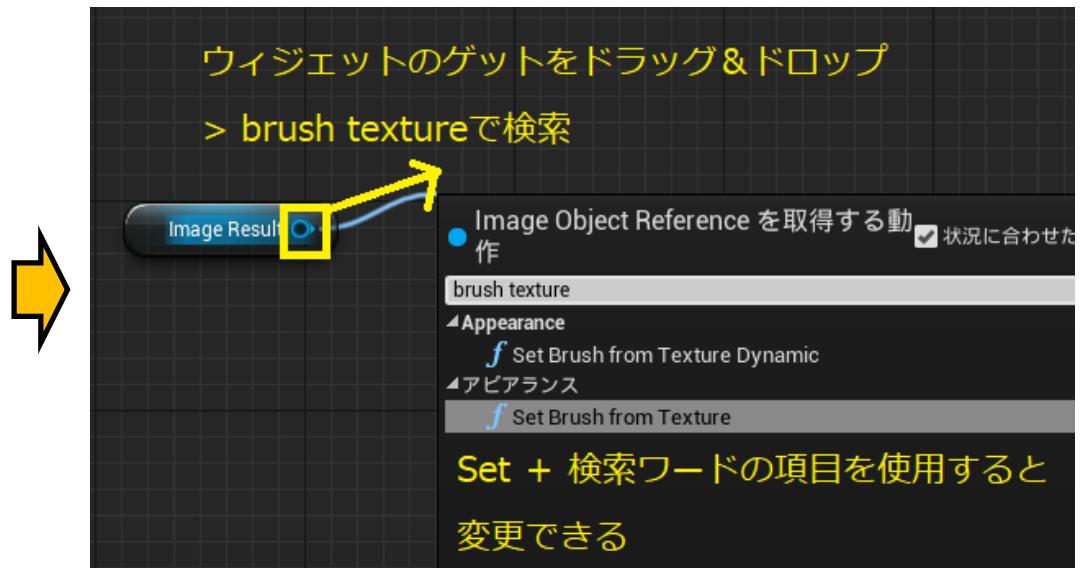


Imageの表示/非表示

ウィジェットの設定を変更したい時のノードを探すコツ



詳細で設定を変更したいプロパティ名や用語を調べる



プロパティを変更したいウィジェットのゲットノードからドラッグ&ドロップ
> 調べたプロパティ名、用語で検索

Set + 検索ワードの項目があれば変更できる

7. UMGからゲームを実行できるようにする

7.1 新規レベル（MathGame）を作成・保存

7.2 WBP_MathGameOutlineを複製して、WBP_MathGameを作成

7.3 背景をアウトラインがない画像に変更

7.4 WBP_MathGameOutlineからWBP_MathGameを表示するようにレベルブループリントを修正

7.5 ウィジェットブループリントの基本機能を実装

 7.5.1 ButtonのClickイベント

 7.5.2 TextのTextプロパティの値を変更

 7.5.3 Imageの画像切り替え

 7.5.4 Imageの表示/非表示

7.6 BP_RandomCalcGameの処理を参考にWBP_MathGameを実装

 7.6.1 BP_RandomCalcGameから変更するポイント

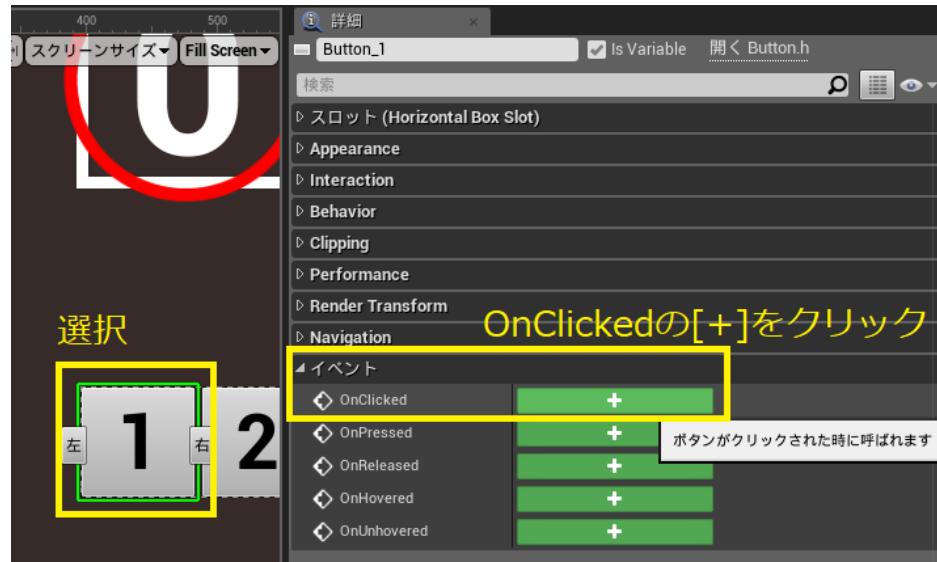
 7.6.2 BP_RandomCalcGameから変数、RandomCalculationを移植

 7.6.3 ボタンのクリックイベントをTextBlock_NumAのTextに設定

 7.6.4 計算とボタンの入力値を判定して結果画像と答えを表示

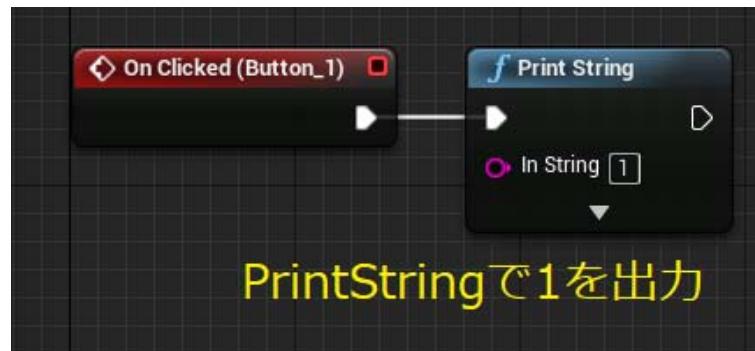
 7.6.5 計算と入力判定のループ化

Button_1のクリックイベントを追加

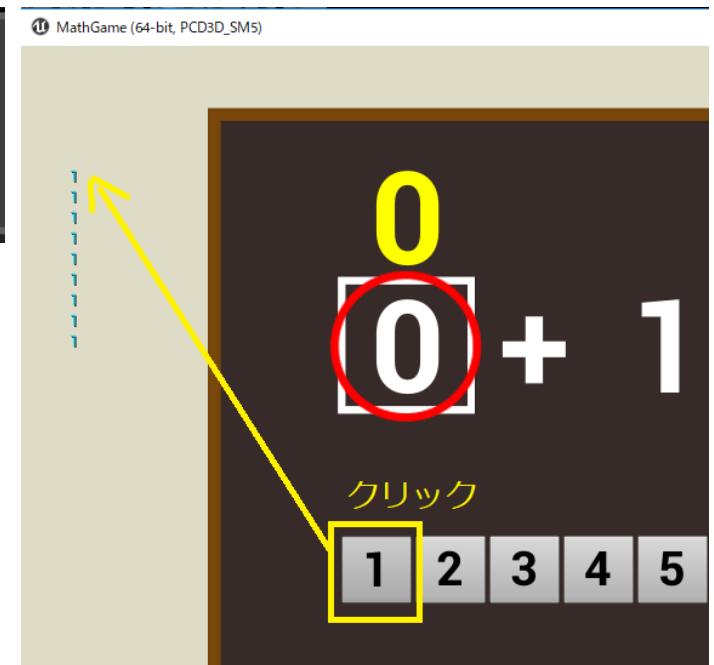


イベントグラフにButton_1のクリックイベントノードが追加される

PrintStringで1を出力するように設定



PrintStringで1を出力するように設定



プレイして確認
1をクリックすると、文字が出力される

7. UMGからゲームを実行できるようにする

7.1 新規レベル（MathGame）を作成・保存

7.2 WBP_MathGameOutlineを複製して、WBP_MathGameを作成

7.3 背景をアウトラインがない画像に変更

7.4 WBP_MathGameOutlineからWBP_MathGameを表示するようにレベルブループリントを修正

7.5 ウィジェットブループリントの基本機能を実装

 7.5.1 ButtonのClickイベント

 7.5.2 TextのTextプロパティの値を変更

 7.5.3 Imageの画像切り替え

 7.5.4 Imageの表示/非表示

7.6 BP_RandomCalcGameの処理を参考にWBP_MathGameを実装

 7.6.1 BP_RandomCalcGameから変更するポイント

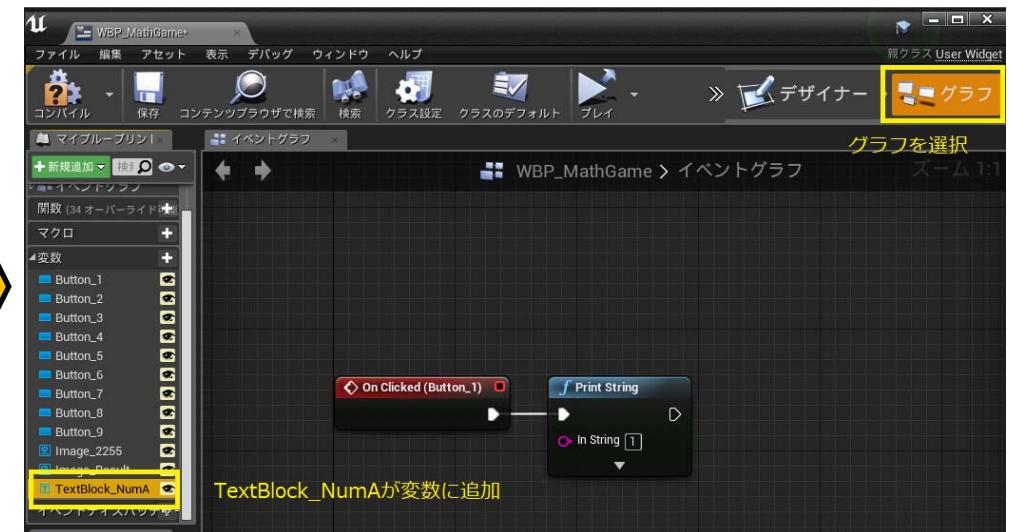
 7.6.2 BP_RandomCalcGameから変数、RandomCalculationを移植

 7.6.3 ボタンのクリックイベントをTextBlock_NumAのTextに設定

 7.6.4 計算とボタンの入力値を判定して結果画像と答えを表示

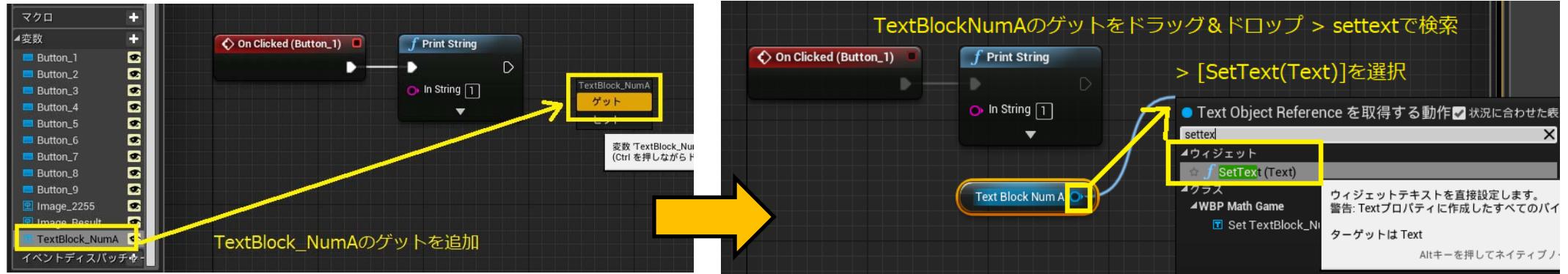
 7.6.5 計算と入力判定のループ化

TextBlock_NumAの変数化



ウィジェットはIs Variableにチェックを入れないと
処理を実装できない
Textは最初にIs Variableにチェックが入っていない

TextBlock_NumAのゲット、 TextのSetText(Text)を追加



TextBlock_NumAのゲットを追加

TextBlock_NumAのゲットをドラッグ&ドロップ
> settextで検索
> [SetText(Text)]を選択

SetText(Text)のIn Textに値を設定



SetText(Text)のIn Textに1を設定

プレイして確認
ボタン：1をクリックすると
テキストがIn Textの値に変わる

7. UMGからゲームを実行できるようにする

7.1 新規レベル（MathGame）を作成・保存

7.2 WBP_MathGameOutlineを複製して、WBP_MathGameを作成

7.3 背景をアウトラインがない画像に変更

7.4 WBP_MathGameOutlineからWBP_MathGameを表示するようにレベルブループリントを修正

7.5 ウィジェットブループリントの基本機能を実装

 7.5.1 ButtonのClickイベント

 7.5.2 TextのTextプロパティの値を変更

 7.5.3 Imageの画像切り替え

 7.5.4 Imageの表示/非表示

7.6 BP_RandomCalcGameの処理を参考にWBP_MathGameを実装

 7.6.1 BP_RandomCalcGameから変更するポイント

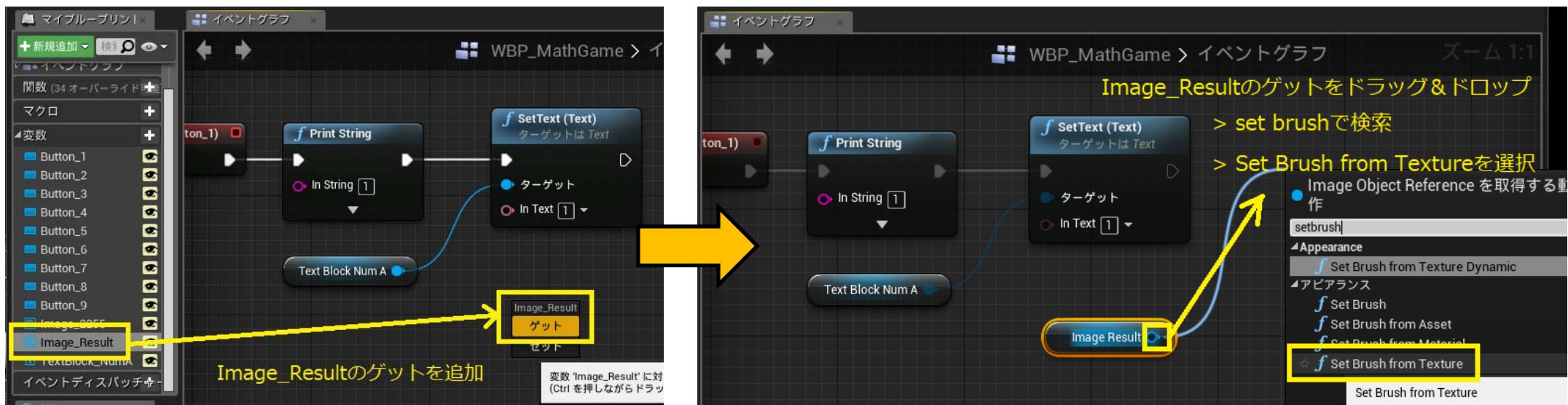
 7.6.2 BP_RandomCalcGameから変数、RandomCalculationを移植

 7.6.3 ボタンのクリックイベントをTextBlock_NumAのTextに設定

 7.6.4 計算とボタンの入力値を判定して結果画像と答えを表示

 7.6.5 計算と入力判定のループ化

Image_Resultのゲットを追加 ImageのSet Brush from Texture

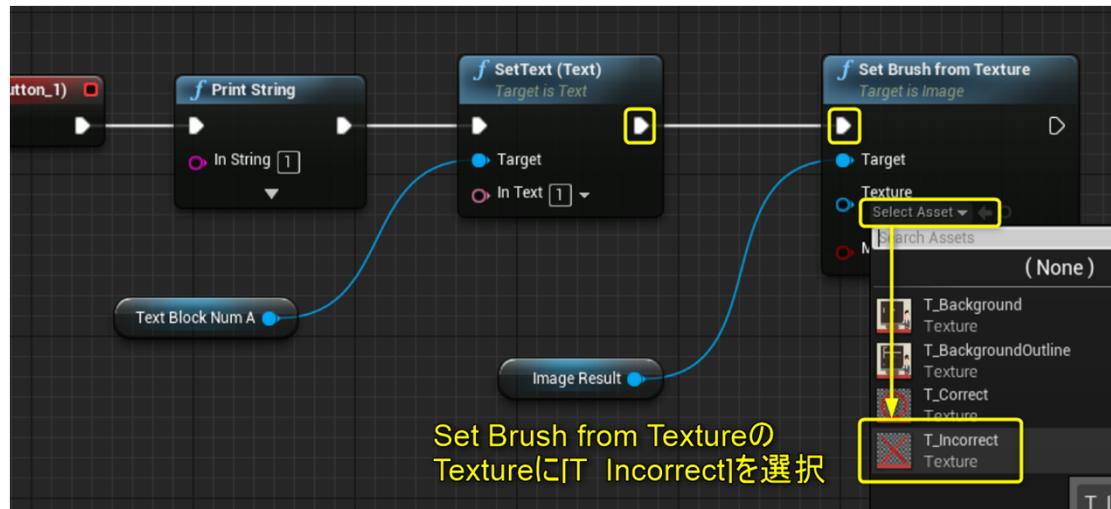


Image_Resultのゲットを追加

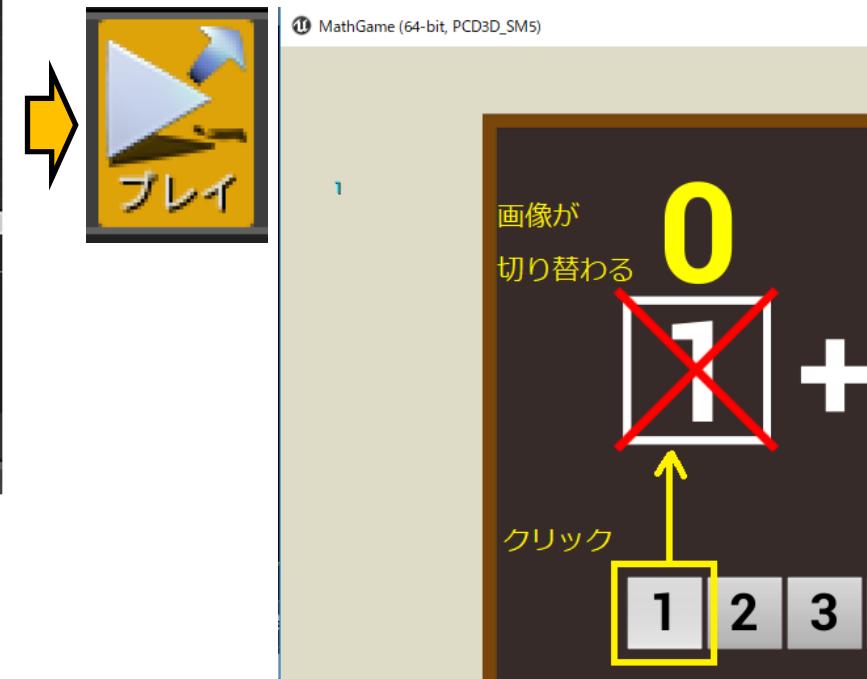
Image_Resultのゲットをドラッグ & ドロップ

- > set brushで検索
- > [Set Brush from Textrure]を選択

Set Brush from Textureの Textureに切り替える画像を設定



Set Brush from Textureの
Textureに[T_Incorrect]を選択



プレイして確認
ボタン1をクリックすると画像が切り替わる

7. UMGからゲームを実行できるようにする

7.1 新規レベル（MathGame）を作成・保存

7.2 WBP_MathGameOutlineを複製して、WBP_MathGameを作成

7.3 背景をアウトラインがない画像に変更

7.4 WBP_MathGameOutlineからWBP_MathGameを表示するようにレベルブループリントを修正

7.5 ウィジェットブループリントの基本機能を実装

 7.5.1 ButtonのClickイベント

 7.5.2 TextのTextプロパティの値を変更

 7.5.3 Imageの画像切り替え

 7.5.4 Imageの表示/非表示

7.6 BP_RandomCalcGameの処理を参考にWBP_MathGameを実装

 7.6.1 BP_RandomCalcGameから変更するポイント

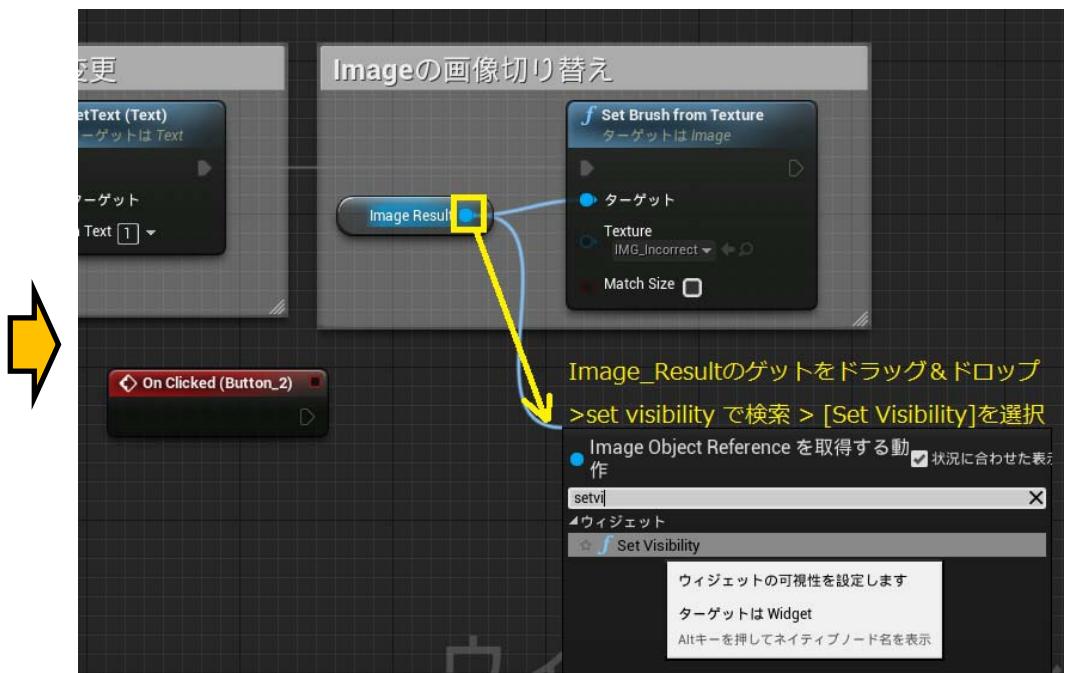
 7.6.2 BP_RandomCalcGameから変数、RandomCalculationを移植

 7.6.3 ボタンのクリックイベントをTextBlock_NumAのTextに設定

 7.6.4 計算とボタンの入力値を判定して結果画像と答えを表示

 7.6.5 計算と入力判定のループ化

ボタン2のクリックイベントを追加
Image_ResultのゲットからSet Visibilityを追加



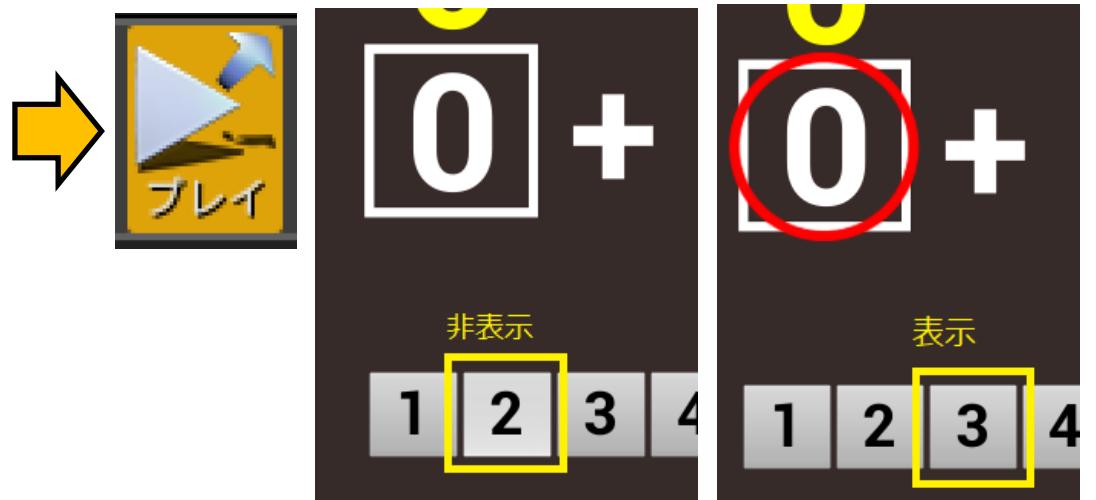
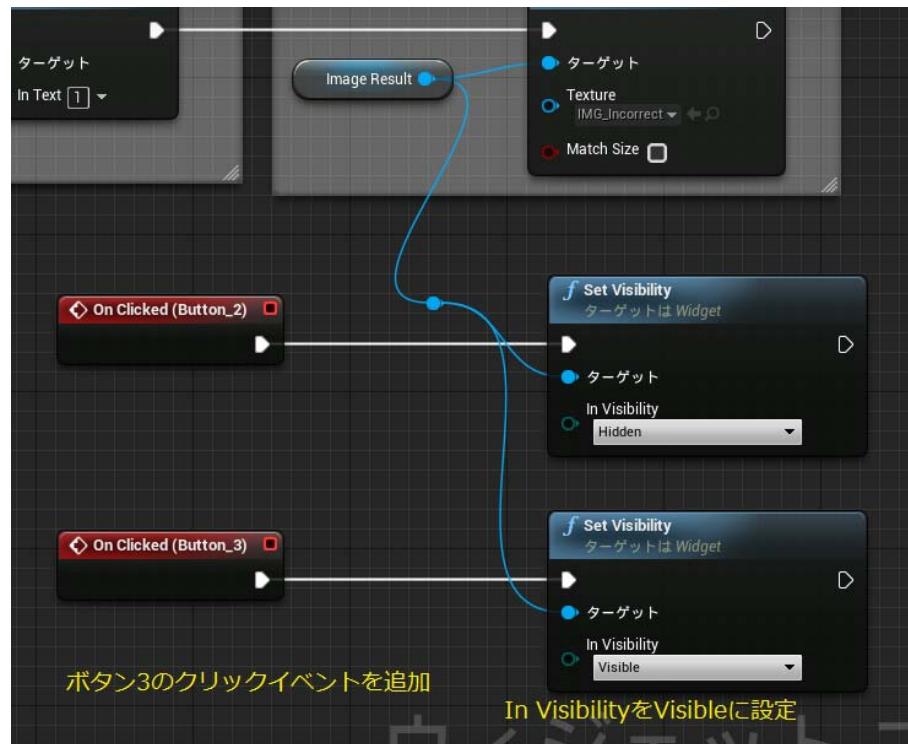
ボタン2のクリックイベントの追加

Image_Resultのゲットをドラッグ&ドロップ
> set visibilityで検索
> [Set Visibility]を選択

Set VisibilityのIn Visibilityを[Hidden]に設定



ボタン3のクリックイベントにSet Visibilityの
In VisibilityをVisibleに設定した処理を実装



プレイして確認
ボタン2(非表示),ボタン3(表示)で
表示/非表示切り替え

ボタン3のクリックイベントを追加
In Visibilityを[Visible]に設定したSet Visibilityを呼び出す

7. UMGからゲームを実行できるようにする

7.1 新規レベル（MathGame）を作成・保存

7.2 WBP_MathGameOutlineを複製して、WBP_MathGameを作成

7.3 背景をアウトラインがない画像に変更

7.4 WBP_MathGameOutlineからWBP_MathGameを表示するようにレベルブループリントを修正

7.5 ウィジェットブループリントの基本機能を実装

 7.5.1 ButtonのClickイベント

 7.5.2 TextのTextプロパティの値を変更

 7.5.3 Imageの画像切り替え

 7.5.4 Imageの表示/非表示

7.6 BP_RandomCalcGameの処理を参考にWBP_MathGameを実装

 7.6.1 BP_RandomCalcGameから変更するポイント

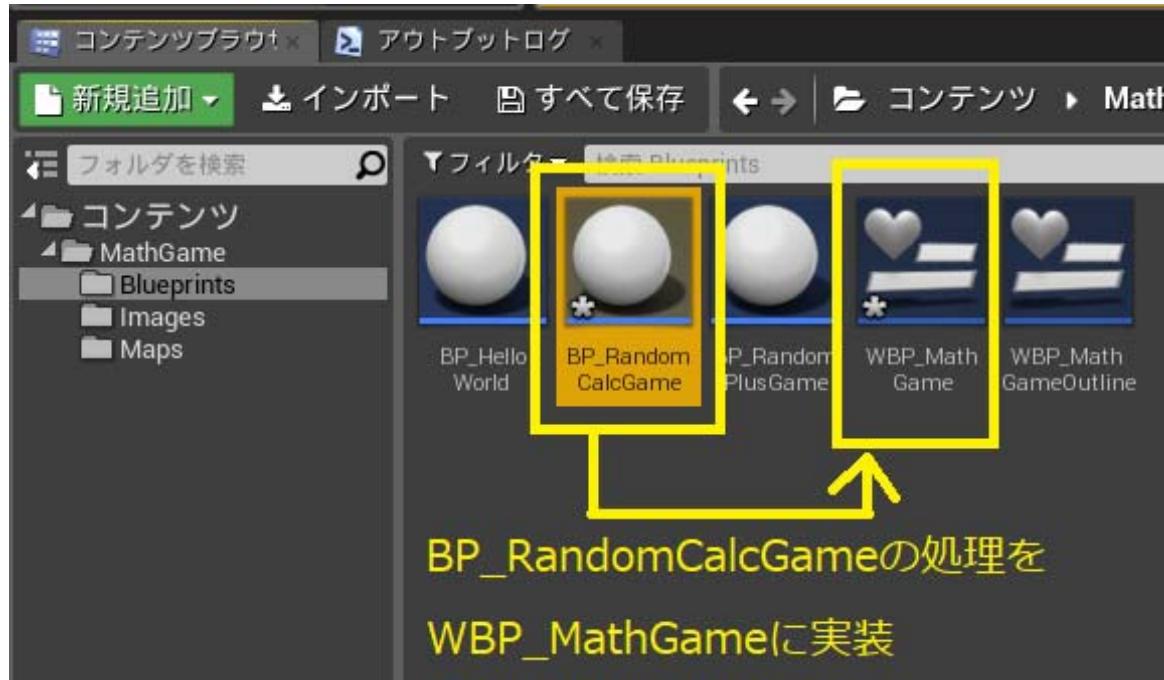
 7.6.2 BP_RandomCalcGameから変数、RandomCalculationを移植

 7.6.3 ボタンのクリックイベントをTextBlock_NumAのTextに設定

 7.6.4 計算とボタンの入力値を判定して結果画像と答えを表示

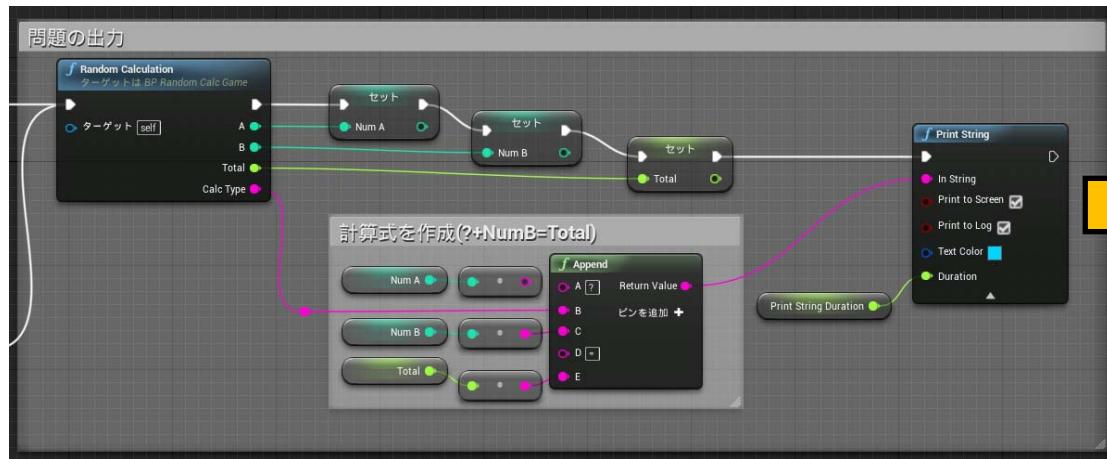
 7.6.5 計算と入力判定のループ化

BP_RandomCalcGameの処理を WBP_MathGameでUMG化する

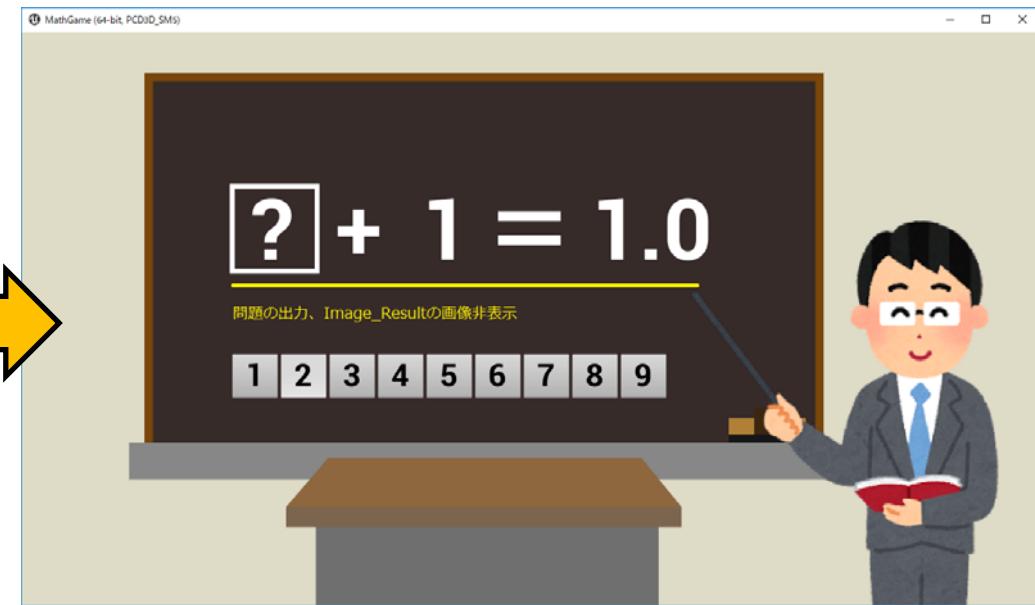


BP_RandomCalcGameはPrint Stringで出力していた処理を
ウィジェットブループリントのウィジェットで表示するように修正する

問題の出力(Print StringからTextで表示)

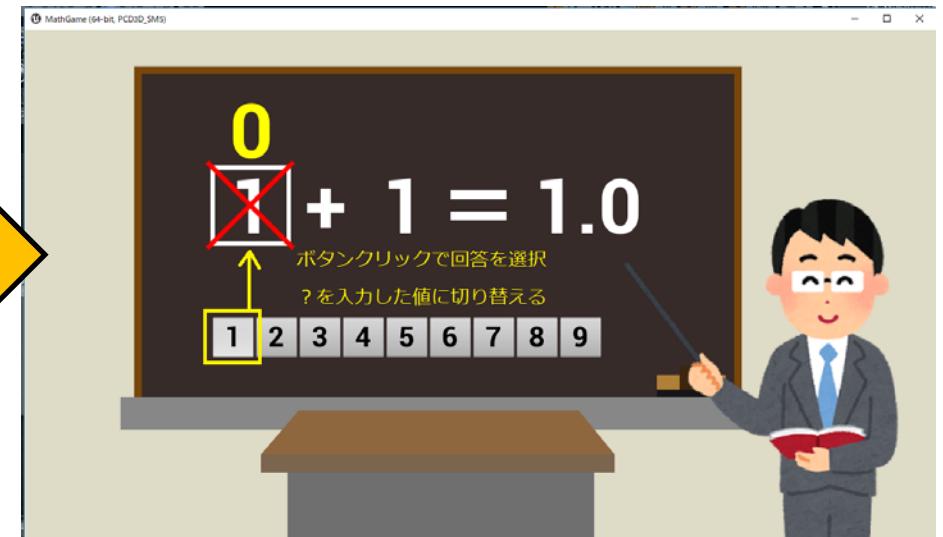
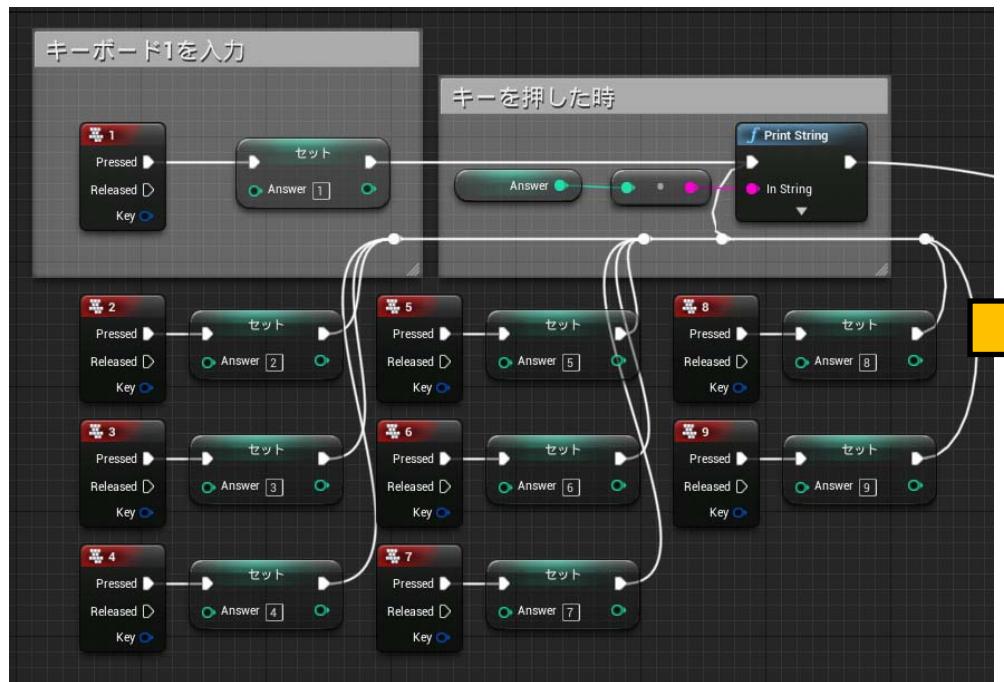


BP_RandomCalcGameの問題出力処理



PrintStringで出力していたが、
Textで計算式を表示するように修正
Image_Resultを非表示にする

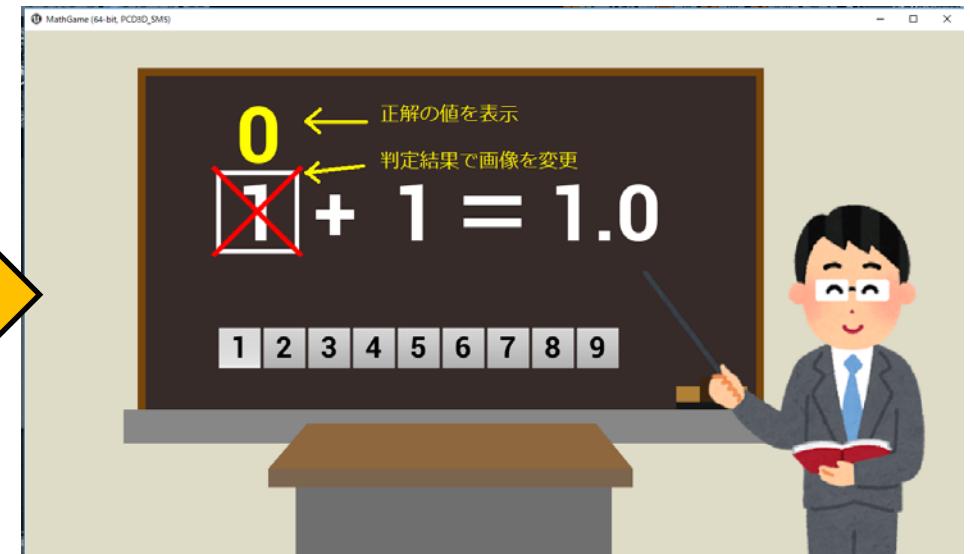
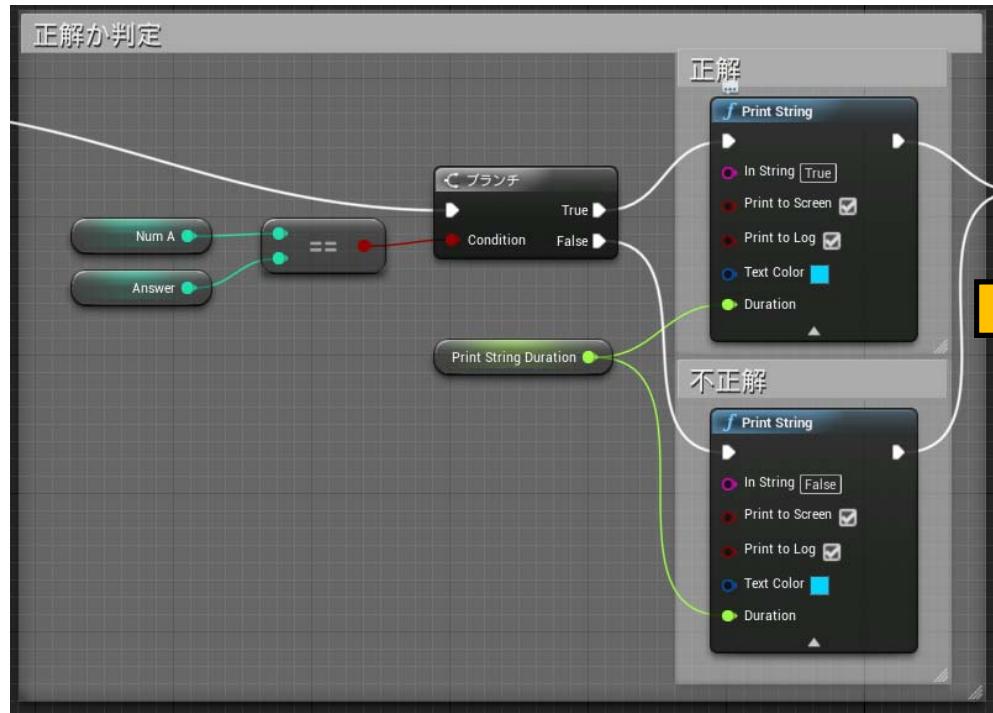
入力(キーボード入力をボタンクリック)



キーボード1~9のイベント処理

ボタンクリックで回答を選択
?を入力した値に切り替える

結果判定(正解の値を表示、結果画像の切替)



ランダム値 NumAの値を表示
判定結果で画像を切替

結果判定処理

入力値とRandomCalculationのランダム値(NumA)を比較

7. UMGからゲームを実行できるようにする

7.1 新規レベル（MathGame）を作成・保存

7.2 WBP_MathGameOutlineを複製して、WBP_MathGameを作成

7.3 背景をアウトラインがない画像に変更

7.4 WBP_MathGameOutlineからWBP_MathGameを表示するようにレベルブループリントを修正

7.5 ウィジェットブループリントの基本機能を実装

 7.5.1 ButtonのClickイベント

 7.5.2 TextのTextプロパティの値を変更

 7.5.3 Imageの画像切り替え

 7.5.4 Imageの表示/非表示

7.6 BP_RandomCalcGameの処理を参考にWBP_MathGameを実装

 7.6.1 BP_RandomCalcGameから変更するポイント

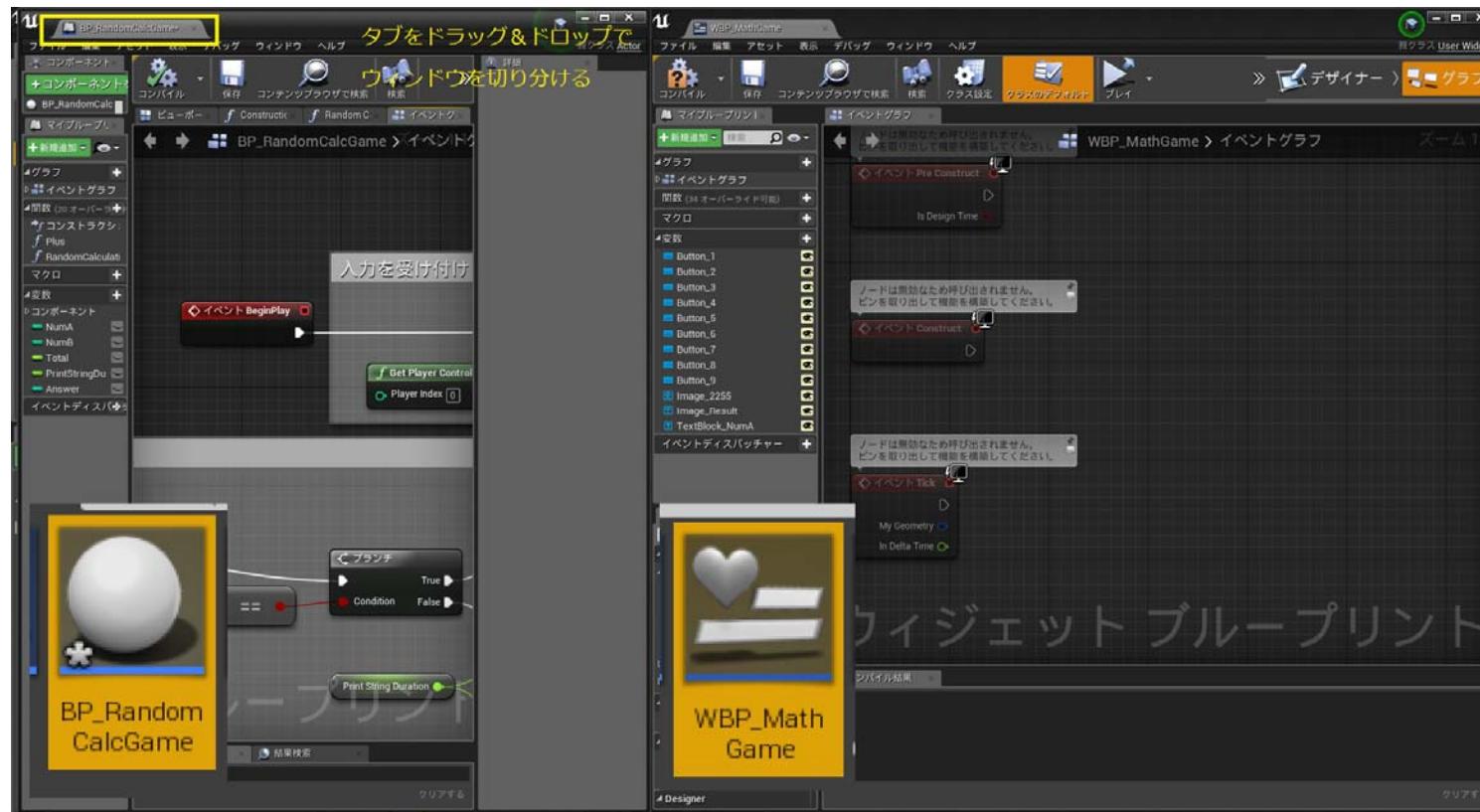
 7.6.2 BP_RandomCalcGameから変数、RandomCalculationを移植

 7.6.3 ボタンのクリックイベントをTextBlock_NumAのTextに設定

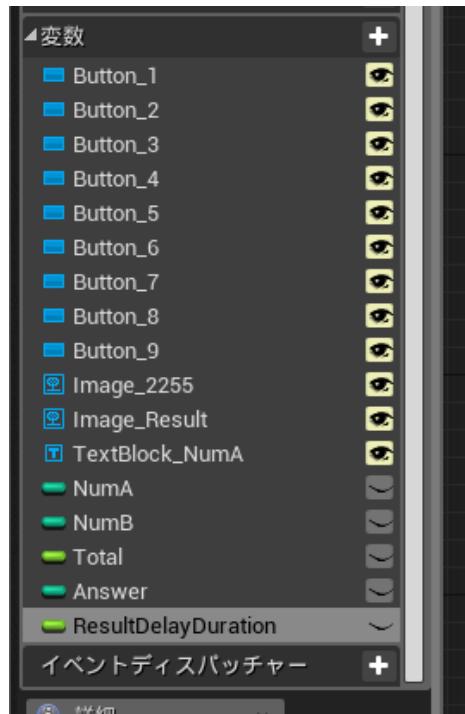
 7.6.4 計算とボタンの入力値を判定して結果画像と答えを表示

 7.6.5 計算と入力判定のループ化

BP_RandomCalcGameとWBP_MathGame を左右に並べながら処理を移し替える

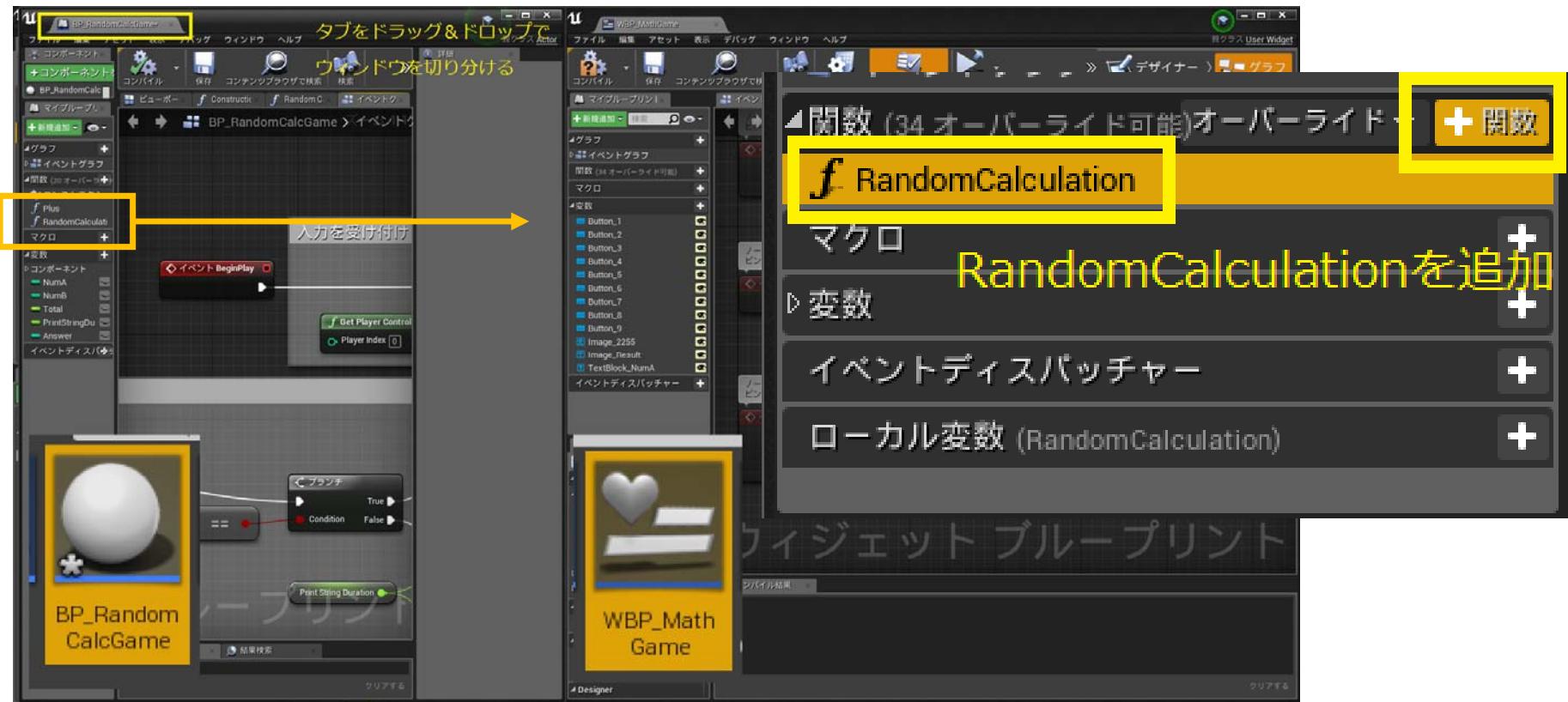


変数の追加



追加する変数名	変数の型	デフォルト値
NumA	Integer	-
NumB	Integer	-
Total	Float	-
Answer	Integer	-
ResultDelayDuration	Float	2

BP_RandomCalcGame から RandomCalculation を移植



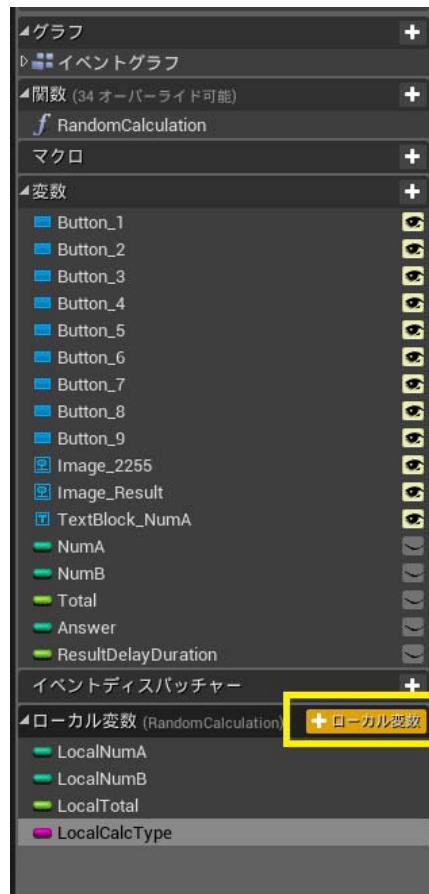
アウトプットを設定する



関数名	Input/Output	パラメータ名	型
RandomCalculation	Output	A	Integer
	Output	B	Integer
	Output	Total	Float
	Output	CalcType	String

アウトプットのパラメータを設定する

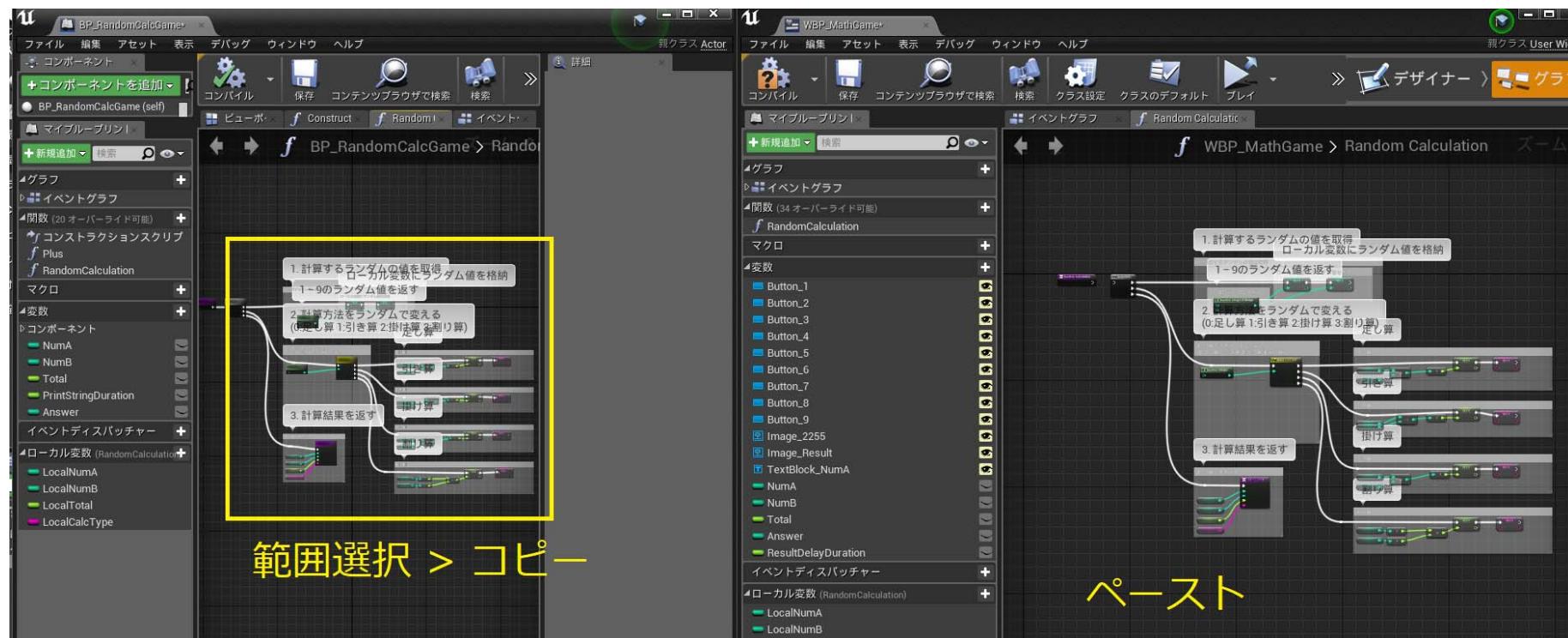
ローカル変数を追加する



ローカル変数名	変数の型
LocalNumA	Integer
LocalNumB	Integer
LocalTotal	Float
LocalCalcType	String

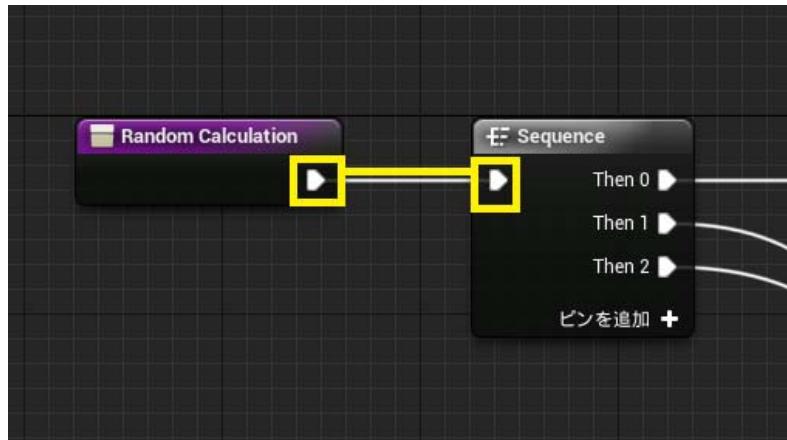
ローカル変数を追加する
(詳細タブで隠れているので、詳細タブを下に下げる)

BP_RandomCalcGameのRandomCalculationの処理を WBP_MathGameのRandomCalculationにコピー & ペースト

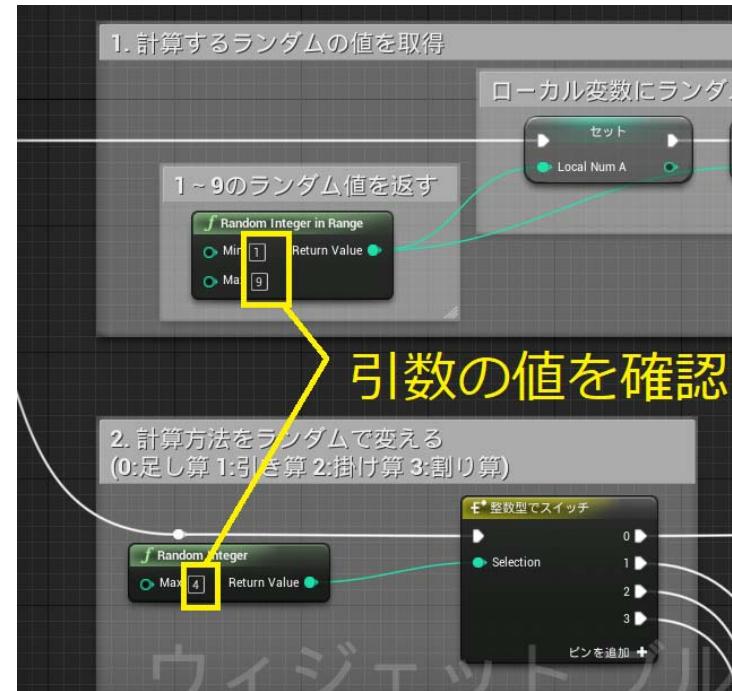


インプット/アウトプット、ローカル変数を同じにしておかないと
処理をコピー & ペーストした際に、処理が外れたり、変数が変わったりエラーの原因になる

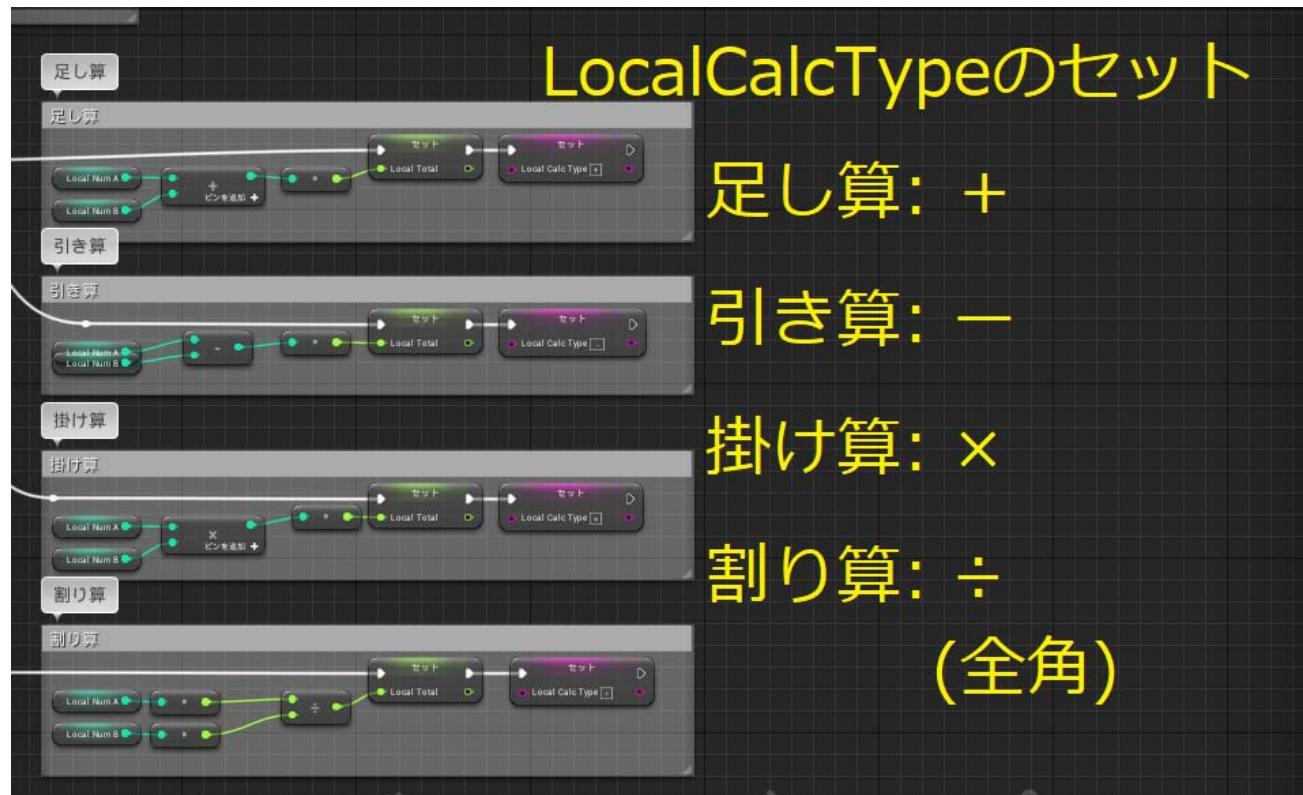
処理が実行されるように繋げなおす
引数の値が変更されていないか確認



処理が実行されるように繋げなおす
つながっていない箇所をつなげる



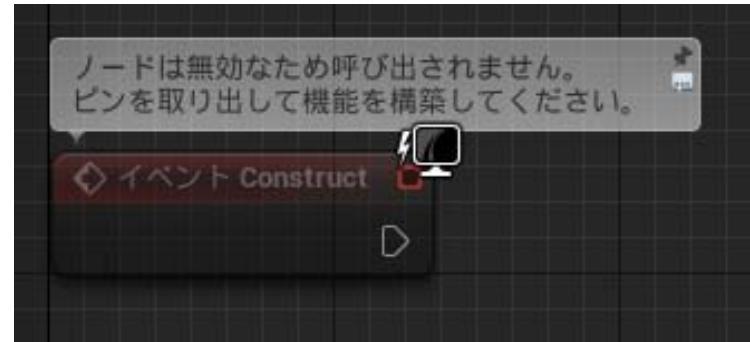
LocalCalcTypeのセットする文字列を全角の計算記号に変更



ウィジェットブループリントでブループリントの
BeginPlayの代わりになるのはイベントConstruct

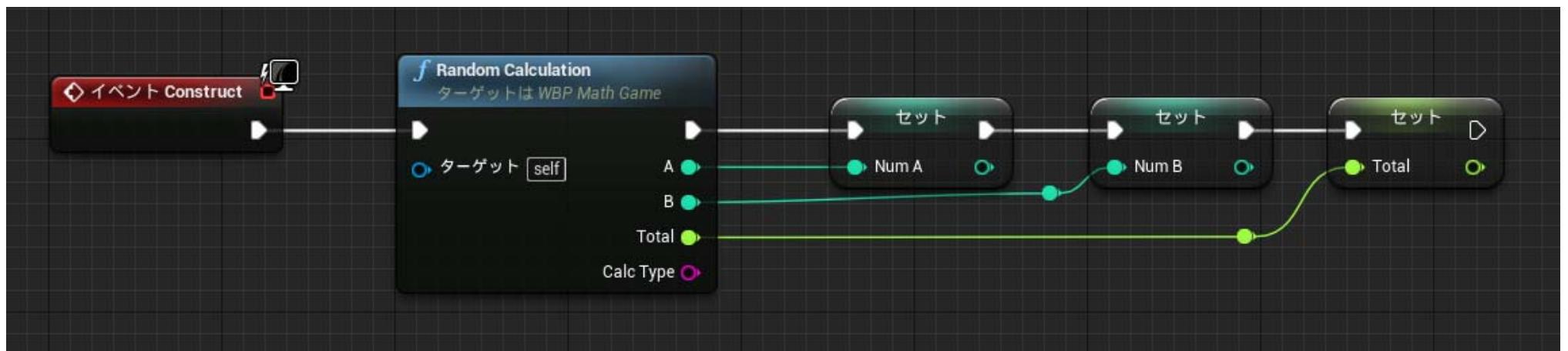


ブループリント(親アクター)では、
イベントBeginPlayから
処理が開始する



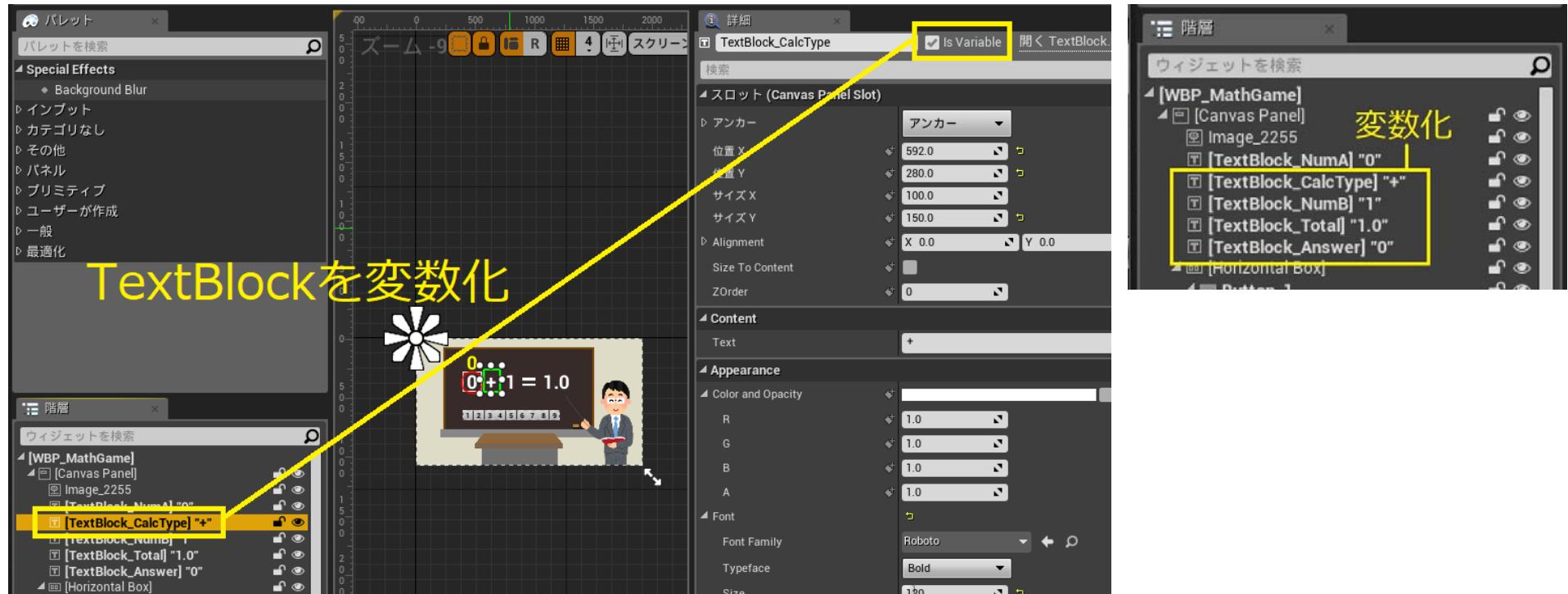
ウィジェットブループリントでは、
BeginPlayの代わりになるのはイベントConstruct

RandomCalculationの結果を変数にセット



RandomCalculationのアウトプットを変数NumA,NumBとTotalに設定する

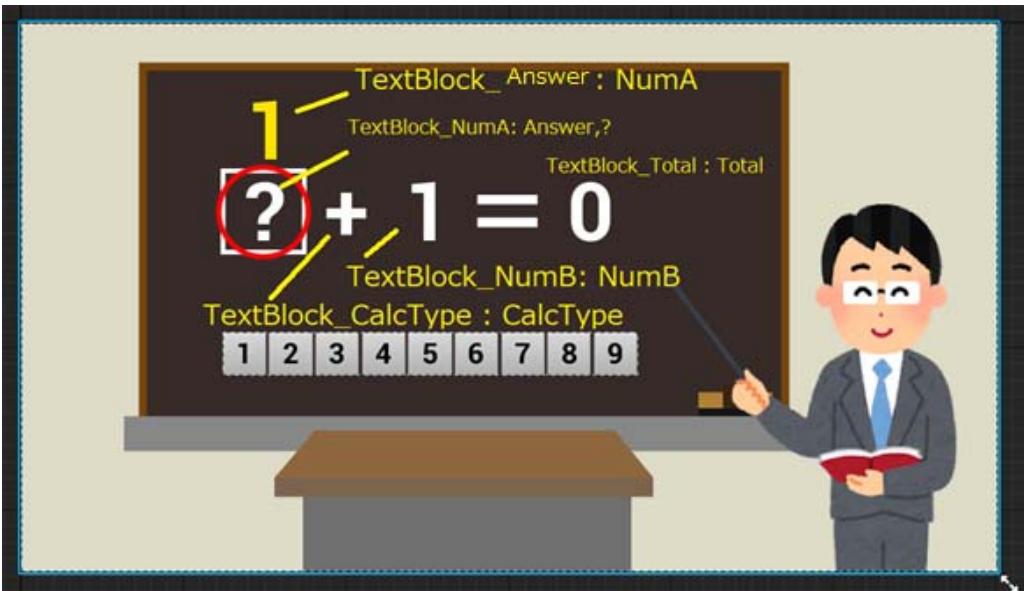
TextBlock～を選択して 全て Is Visibilityにチェック



計算式の表示をするTextウィジェットを全て変数化する

計算式の作成 2

TextBlockに表示する変数、文字列

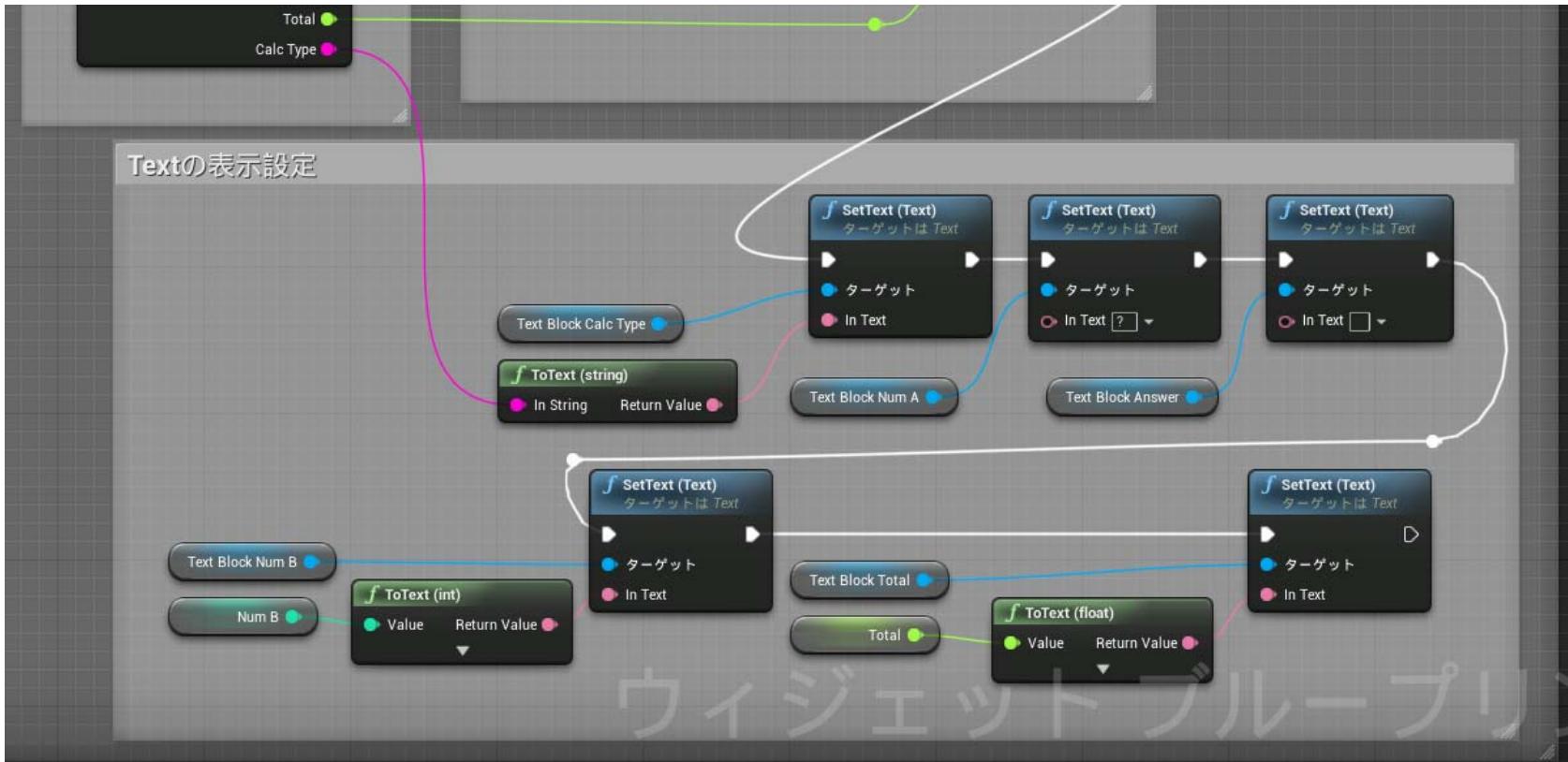


Textには変数名または文字列を表示

変数名	計算式表示	結果表示
TextBlock_Answer	""(空白)	変数 NumA
TextBlock_NumA	"?“	変数 Answer
TextBlock_CalcType	アウトプット CalcType	アウトプット CalcType
TextBlock_NumB	変数 NumB	変数 NumB
TextBlock_Total	変数 Total	変数 Total

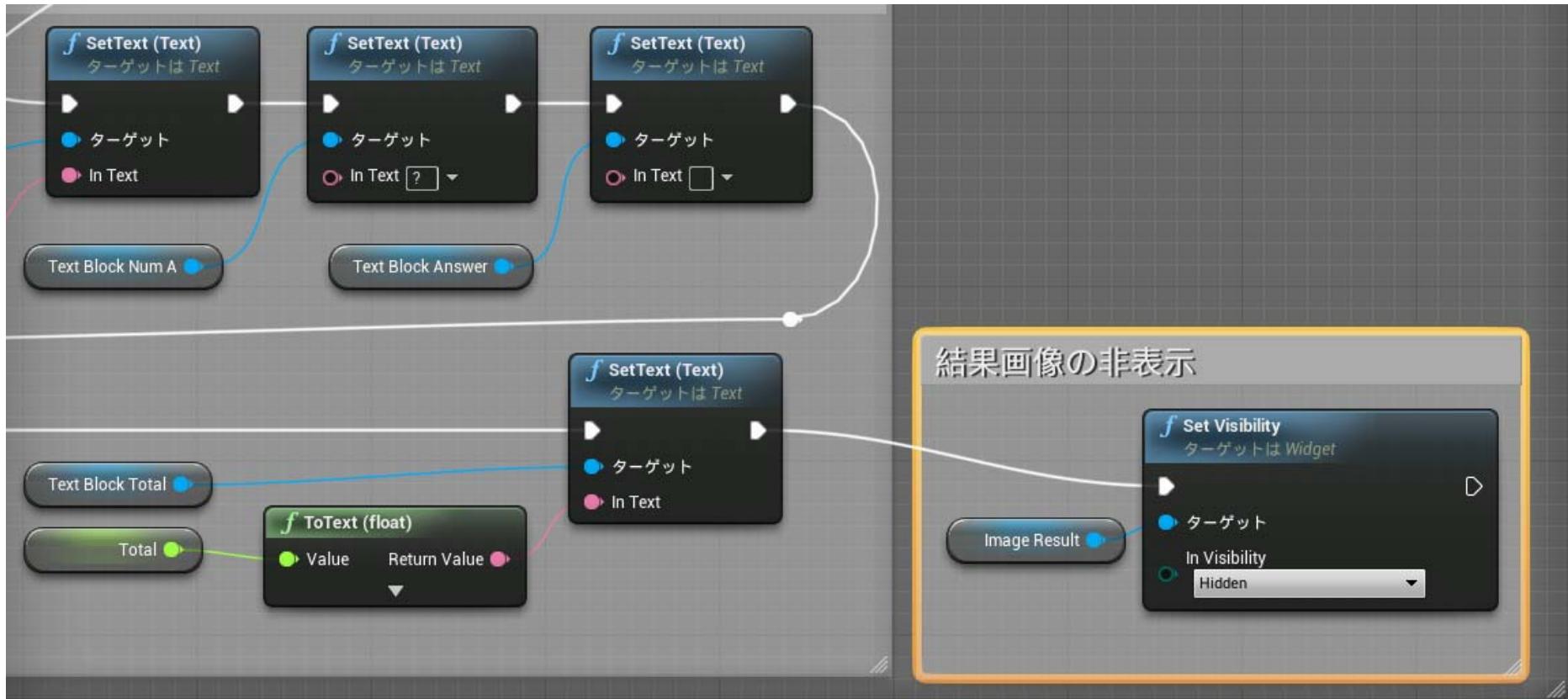
“”は文字列

Textの値を変更して計算式を作成



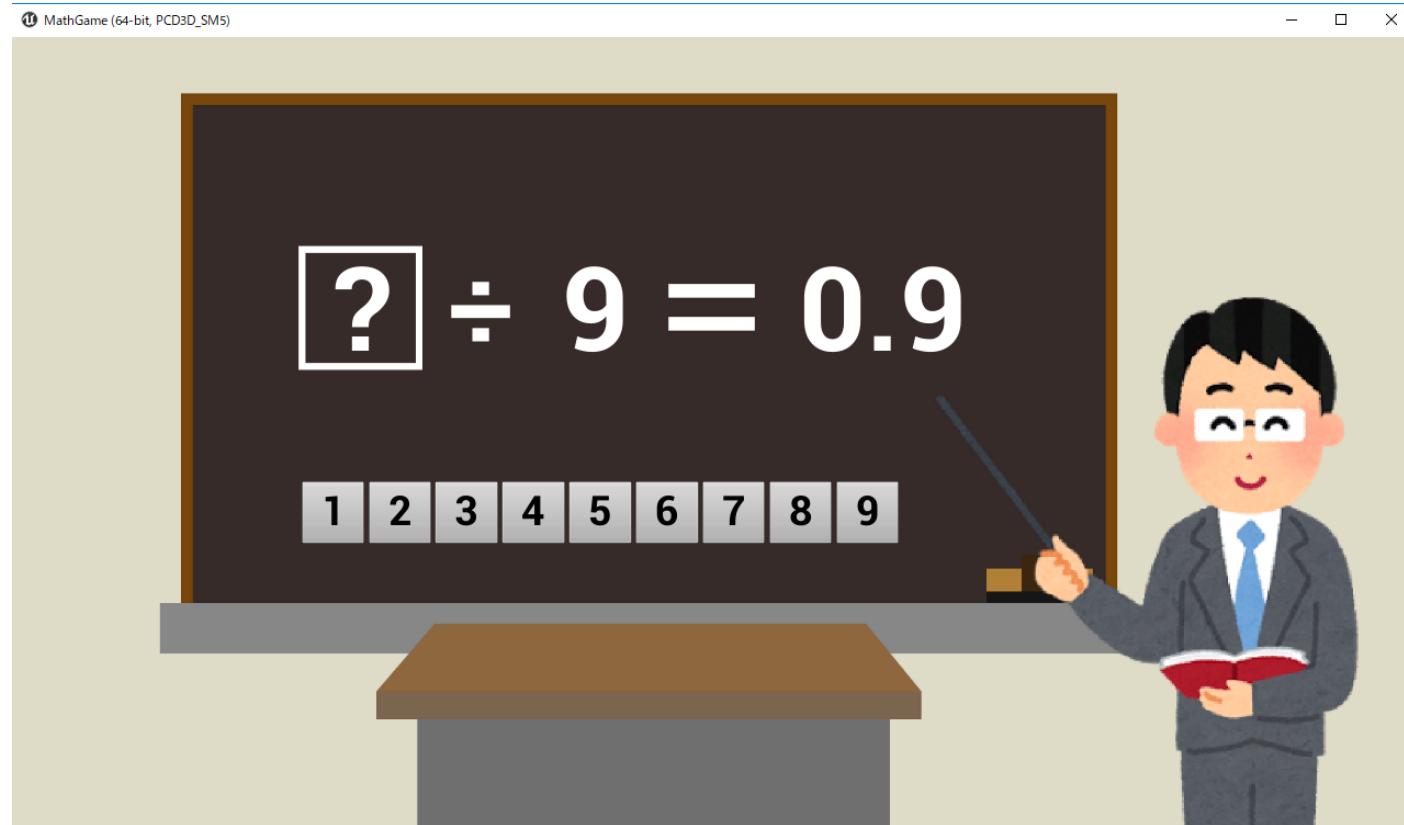
TextウィジェットのTextプロパティをSetTextで変更して計算式を作成する

結果画像の非表示



結果画像を SetVisibility で非表示に設定する

計算式が表示されることを確認



7. UMGからゲームを実行できるようにする

7.1 新規レベル（MathGame）を作成・保存

7.2 WBP_MathGameOutlineを複製して、WBP_MathGameを作成

7.3 背景をアウトラインがない画像に変更

7.4 WBP_MathGameOutlineからWBP_MathGameを表示するようにレベルブループリントを修正

7.5 ウィジェットブループリントの基本機能を実装

 7.5.1 ButtonのClickイベント

 7.5.2 TextのTextプロパティの値を変更

 7.5.3 Imageの画像切り替え

 7.5.4 Imageの表示/非表示

7.6 BP_RandomCalcGameの処理を参考にWBP_MathGameを実装

 7.6.1 BP_RandomCalcGameから変更するポイント

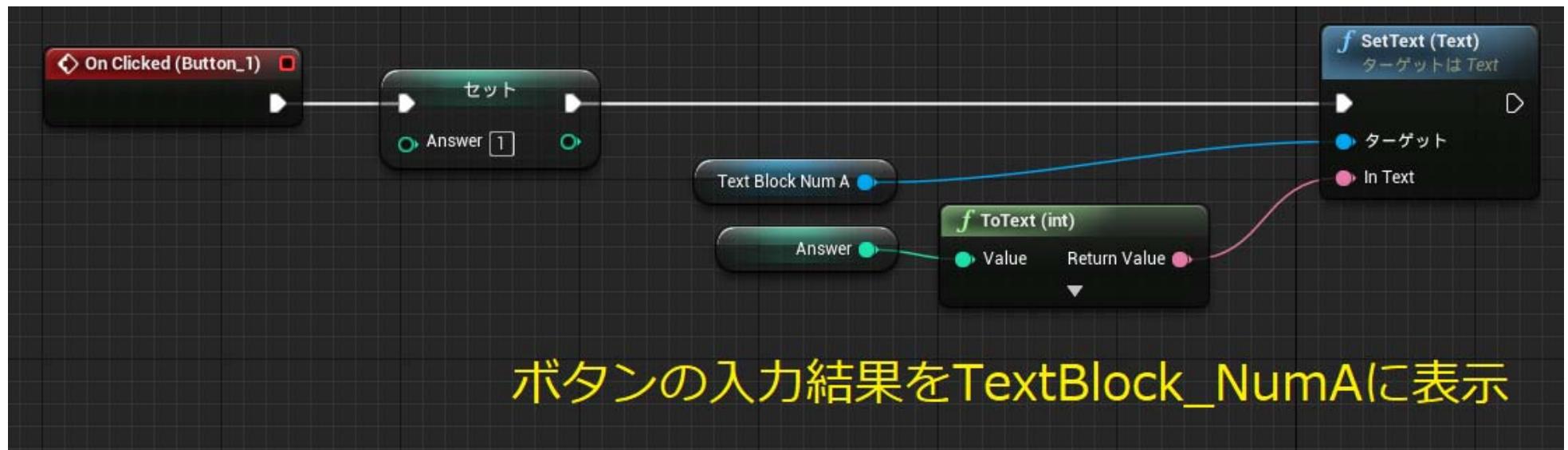
 7.6.2 BP_RandomCalcGameから変数、RandomCalculationを移植

 7.6.3 ボタンのクリックイベントをTextBlock_NumAのTextに設定

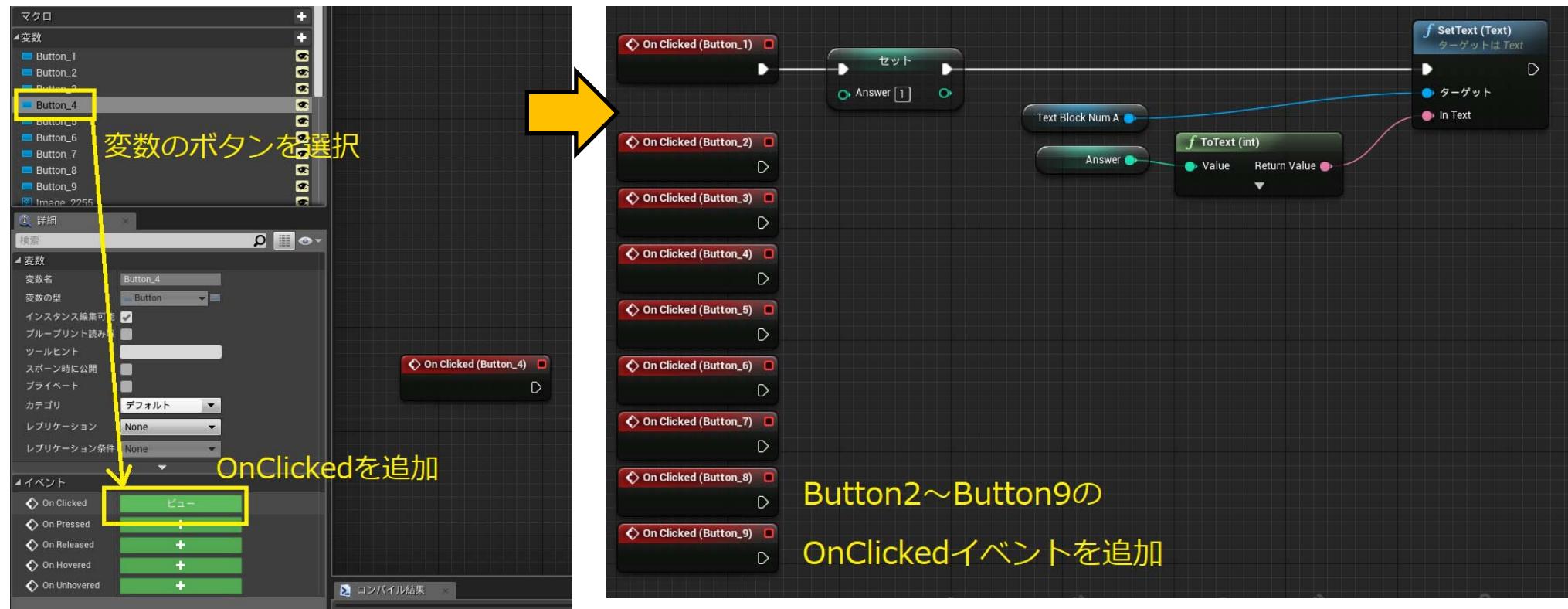
 7.6.4 計算とボタンの入力値を判定して結果画像と答えを表示

 7.6.5 計算と入力判定のループ化

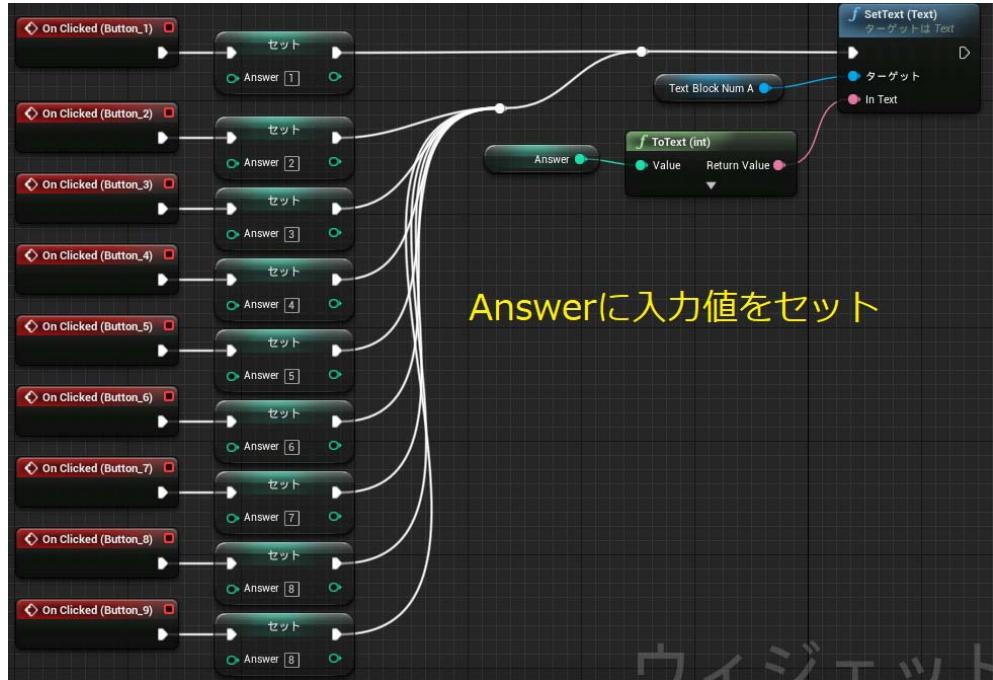
ボタンの入力結果をTextBlock_NumAに表示



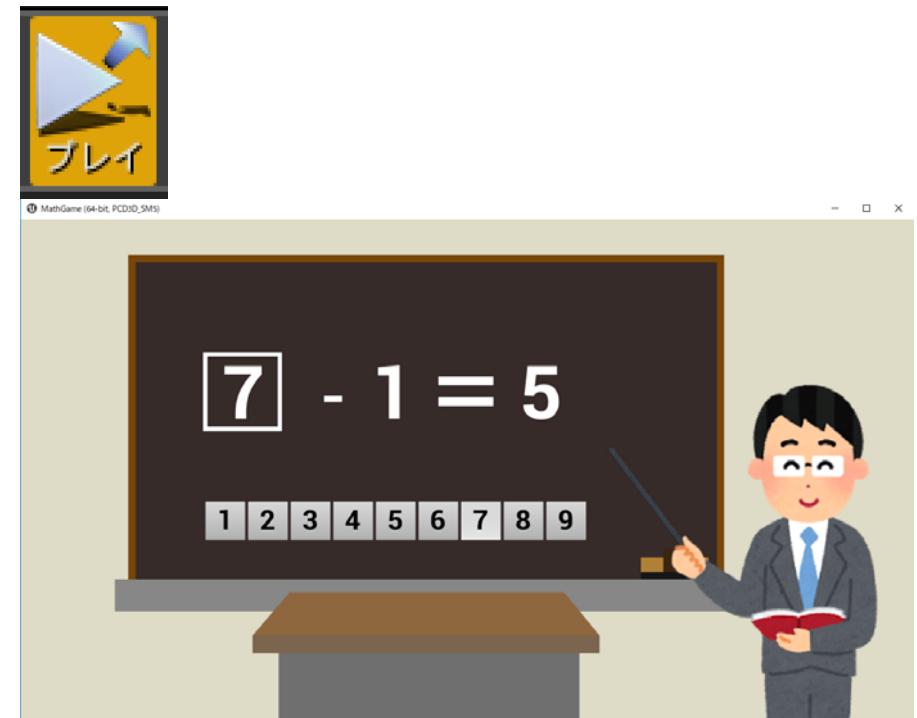
Button_2～Button_9の イベントOnClickedを追加



Answerに入力値をセット



Button_2～Button_9までのボタンの入力値を
Answerにセット



プレイして確認
ボタンをクリックすると数値が変わることを確認

7. UMGからゲームを実行できるようにする

7.1 新規レベル（MathGame）を作成・保存

7.2 WBP_MathGameOutlineを複製して、WBP_MathGameを作成

7.3 背景をアウトラインがない画像に変更

7.4 WBP_MathGameOutlineからWBP_MathGameを表示するようにレベルブループリントを修正

7.5 ウィジェットブループリントの基本機能を実装

 7.5.1 ButtonのClickイベント

 7.5.2 TextのTextプロパティの値を変更

 7.5.3 Imageの画像切り替え

 7.5.4 Imageの表示/非表示

7.6 BP_RandomCalcGameの処理を参考にWBP_MathGameを実装

 7.6.1 BP_RandomCalcGameから変更するポイント

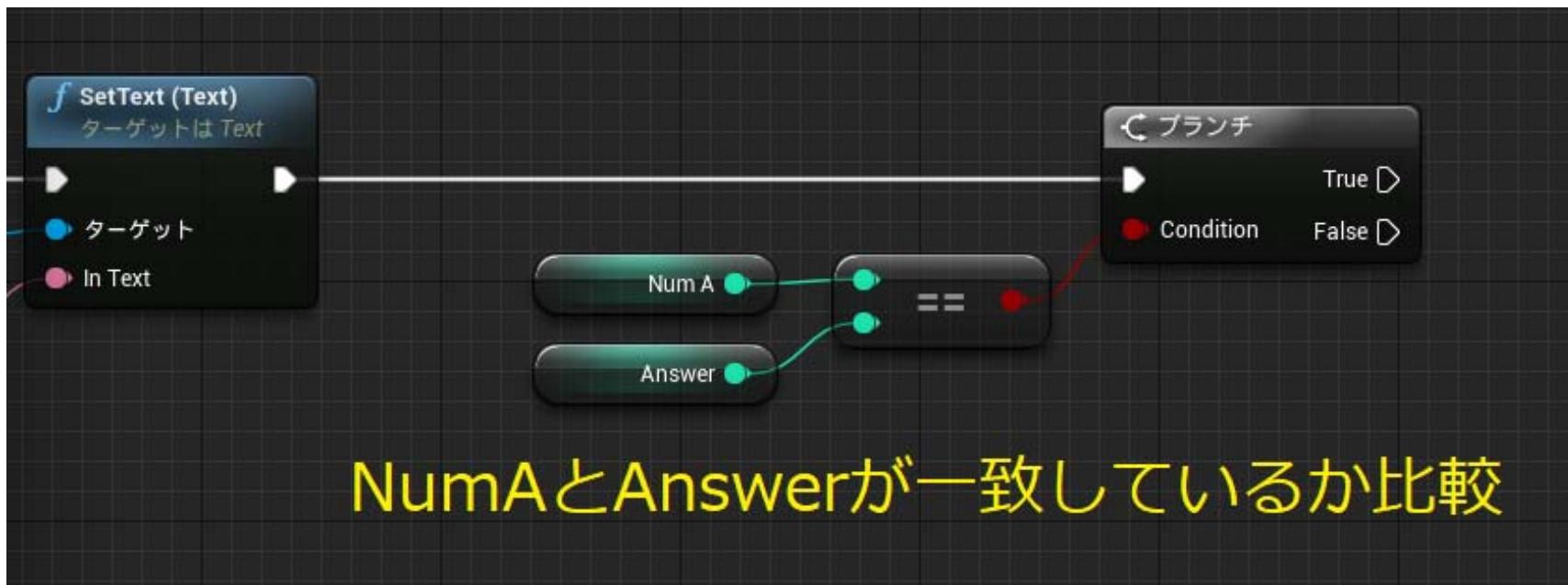
 7.6.2 BP_RandomCalcGameから変数、RandomCalculationを移植

 7.6.3 ボタンのクリックイベントをTextBlock_NumAのTextに設定

 7.6.4 計算とボタンの入力値を判定して結果画像と答えを表示

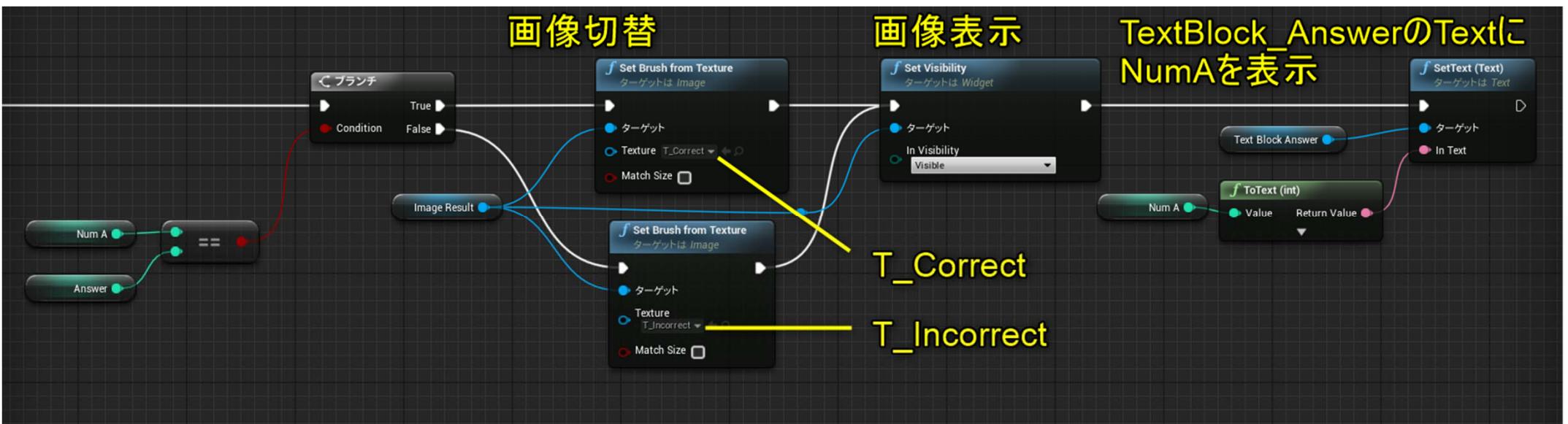
 7.6.5 計算と入力判定のループ化

NumAとAnswerが一致しているか比較

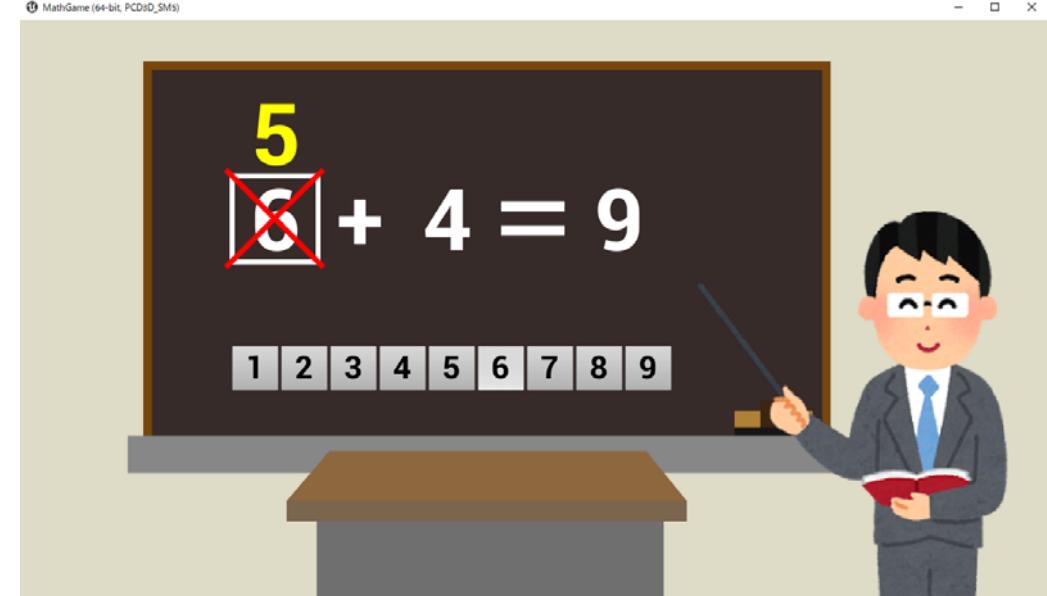
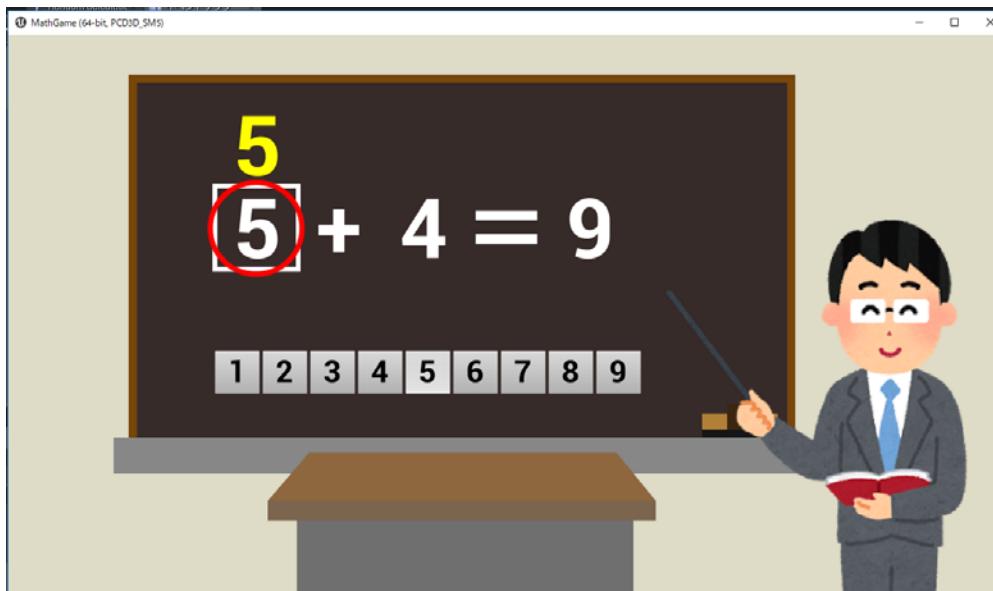


NumAとAnswerが一致しているか比較

画像の切替>画像表示>Answerを表示



プレイして確認



正解/不正解で結果が変わる
ボタンが何度も押せてしまう

7. UMGからゲームを実行できるようにする

7.1 新規レベル（MathGame）を作成・保存

7.2 WBP_MathGameOutlineを複製して、WBP_MathGameを作成

7.3 背景をアウトラインがない画像に変更

7.4 WBP_MathGameOutlineからWBP_MathGameを表示するようにレベルブループリントを修正

7.5 ウィジェットブループリントの基本機能を実装

 7.5.1 ButtonのClickイベント

 7.5.2 TextのTextプロパティの値を変更

 7.5.3 Imageの画像切り替え

 7.5.4 Imageの表示/非表示

7.6 BP_RandomCalcGameの処理を参考にWBP_MathGameを実装

 7.6.1 BP_RandomCalcGameから変更するポイント

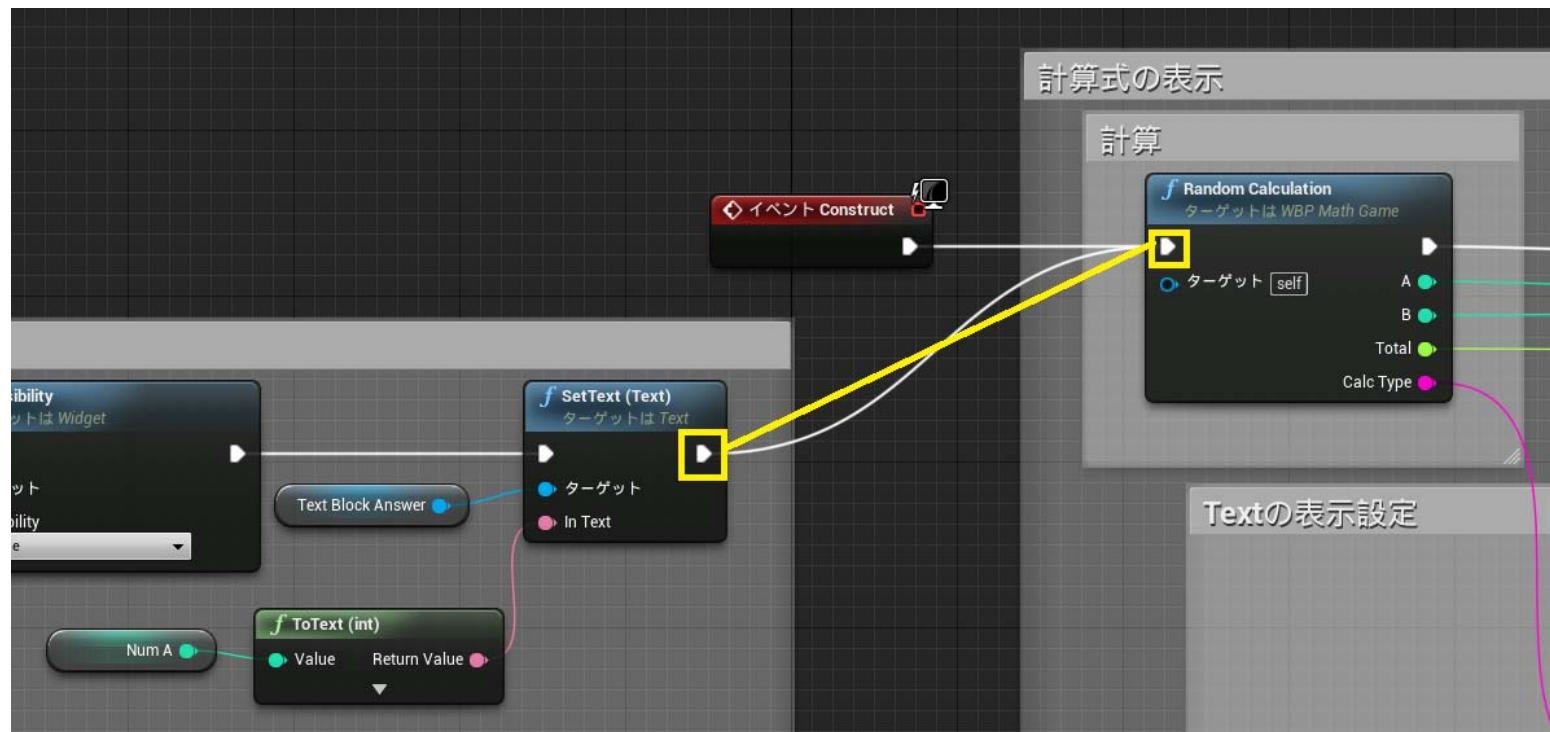
 7.6.2 BP_RandomCalcGameから変数、RandomCalculationを移植

 7.6.3 ボタンのクリックイベントをTextBlock_NumAのTextに設定

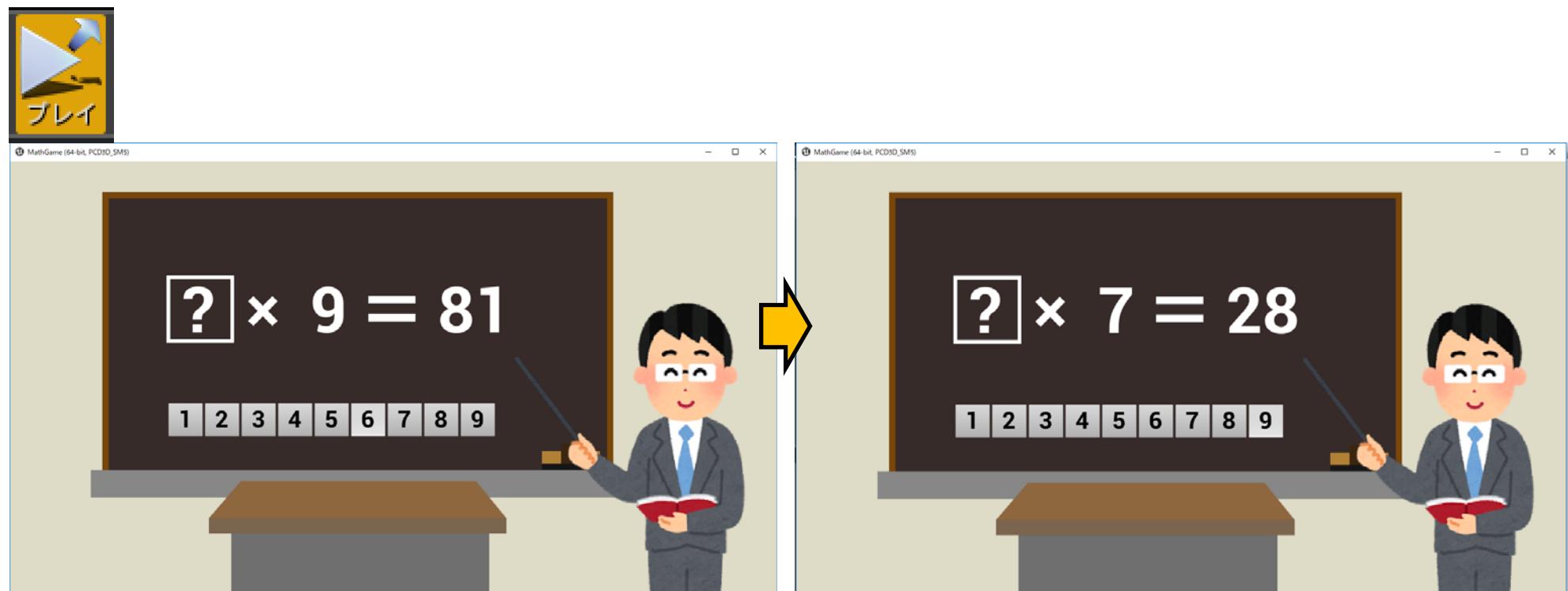
 7.6.4 計算とボタンの入力値を判定して結果画像と答えを表示

 7.6.5 計算と入力判定のループ化

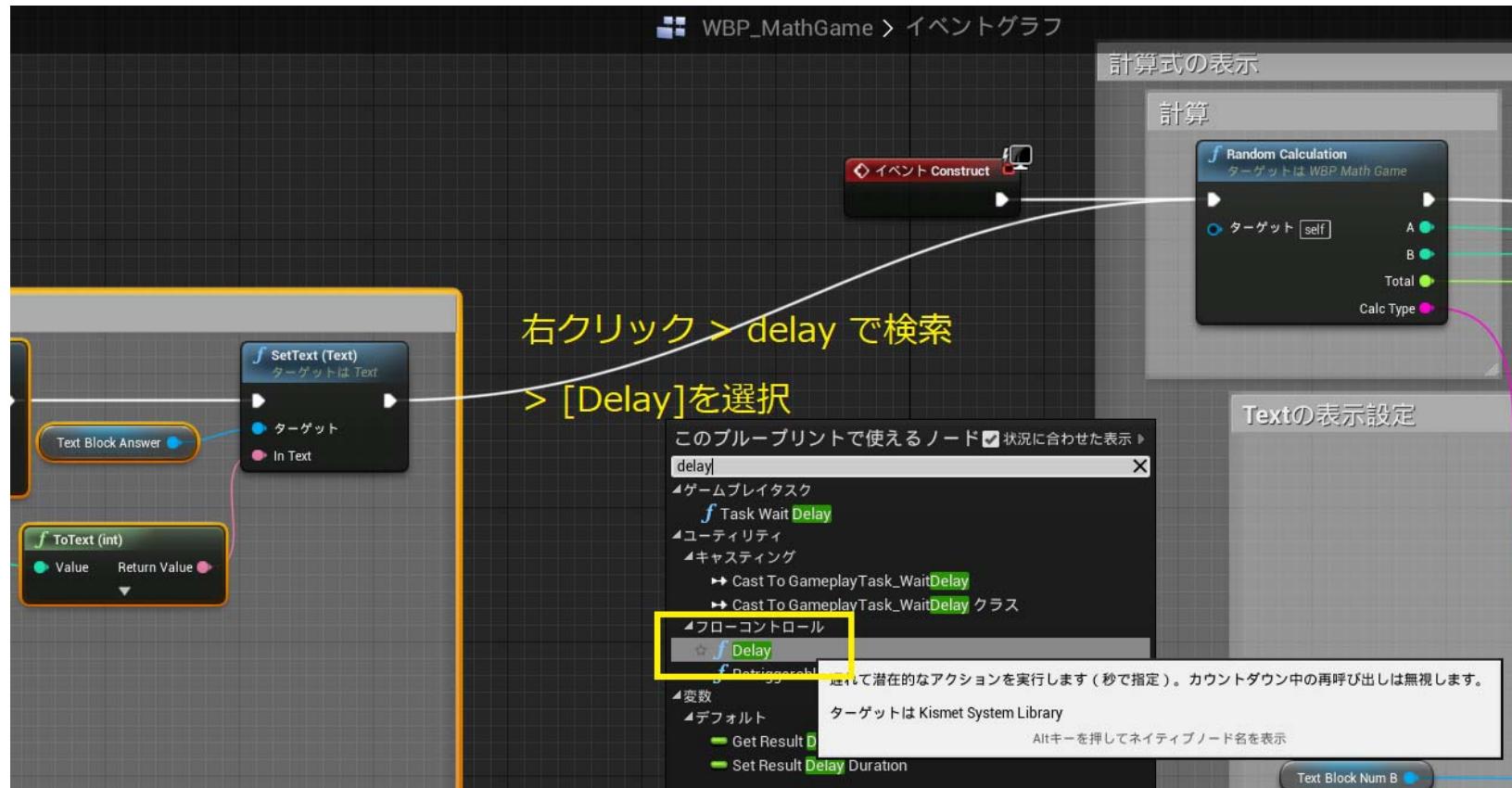
判定結果を表示した後に
RandomCalculationを実行



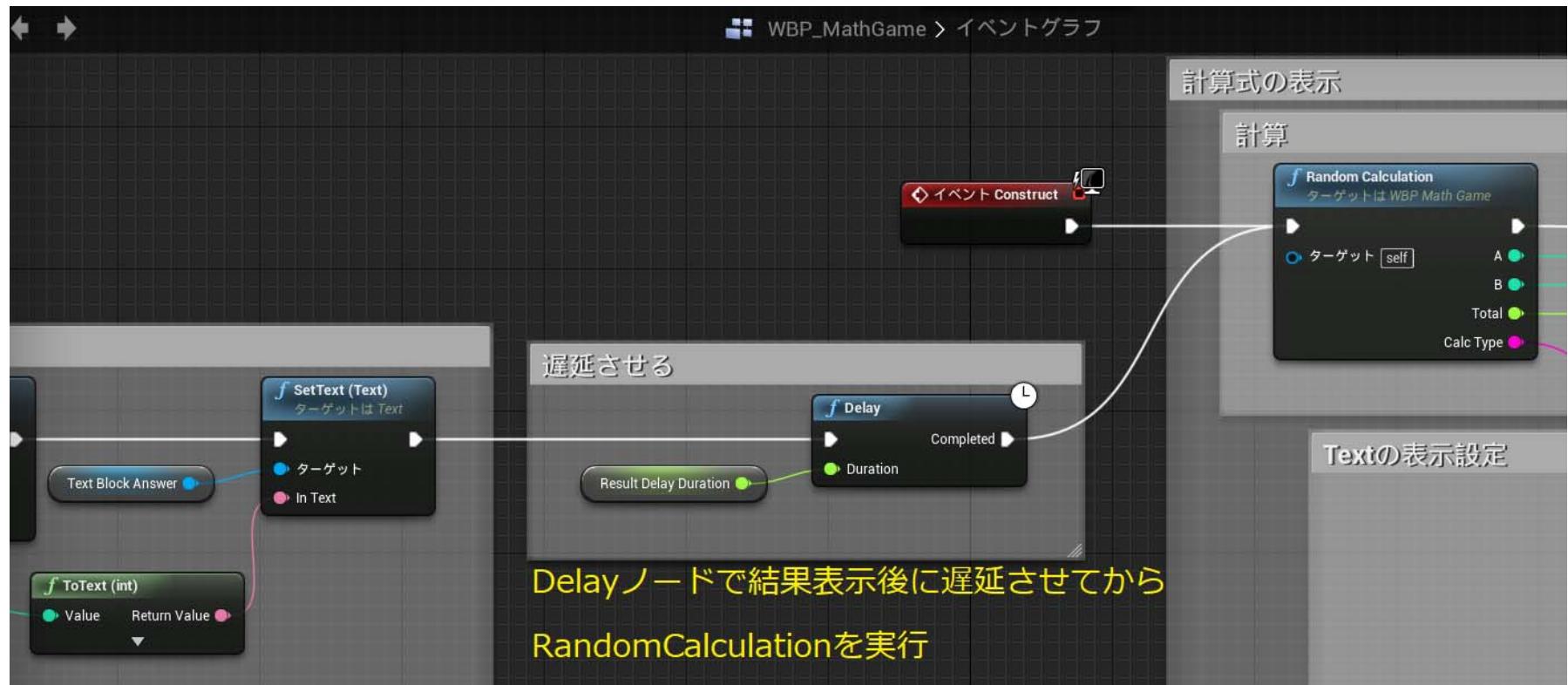
ボタンをクリックすると計算式が変わ
るが、結果が表示されない



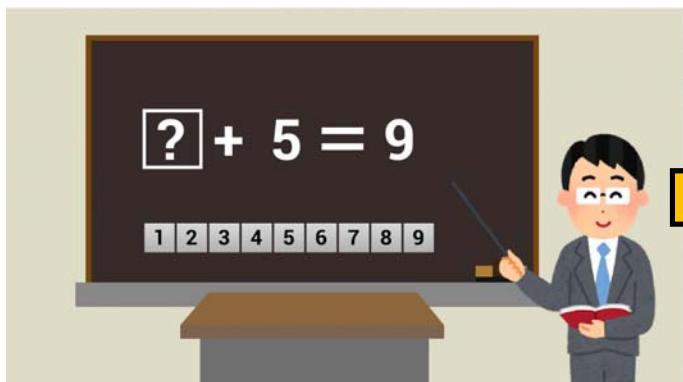
Delayを追加



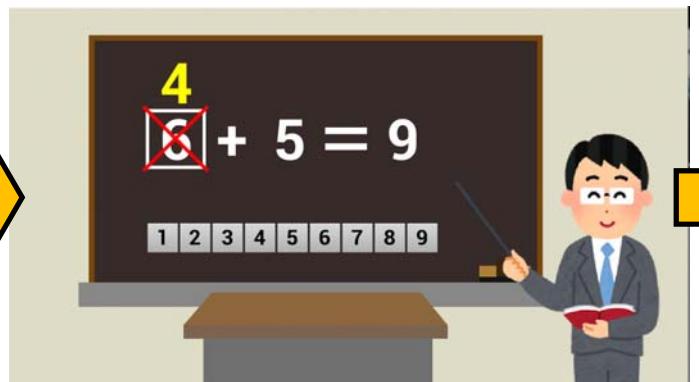
Delayノードで結果表示後に遅延させてからRandomCalculationを実行



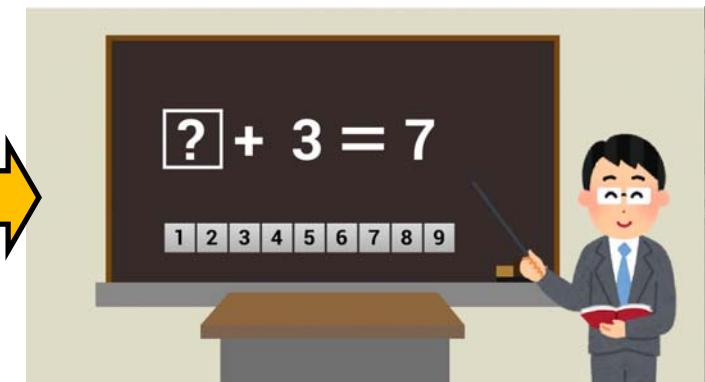
ボタンクリック後、結果を表示して、計算式を表示



ボタンをクリック



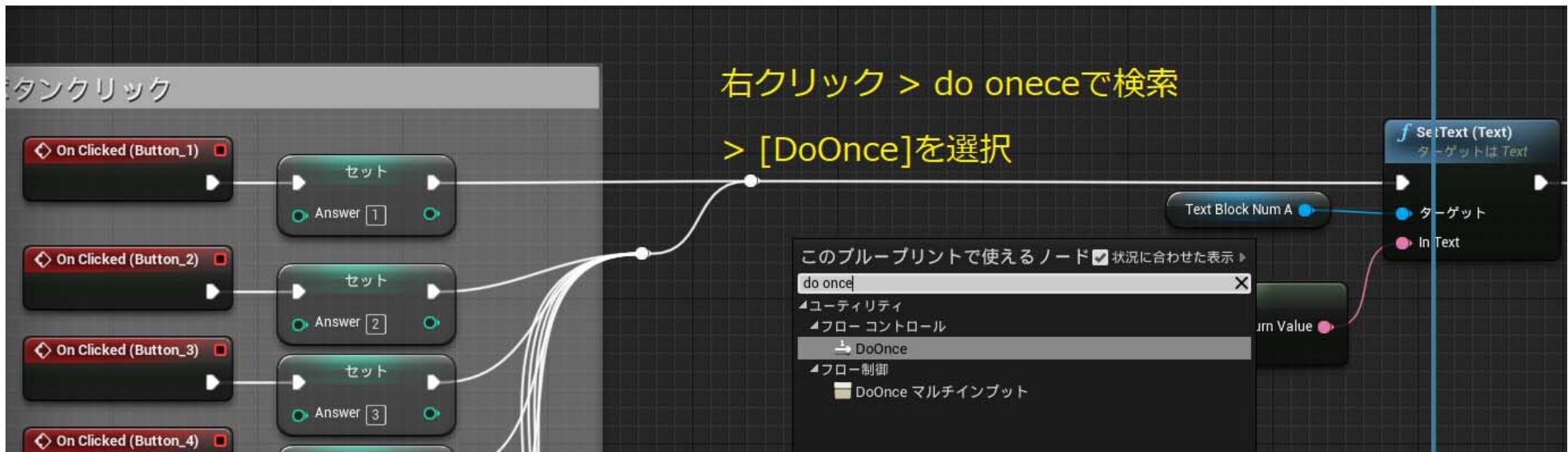
結果がしばらく表示される



計算式の表示

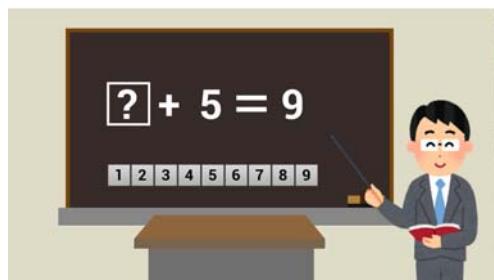
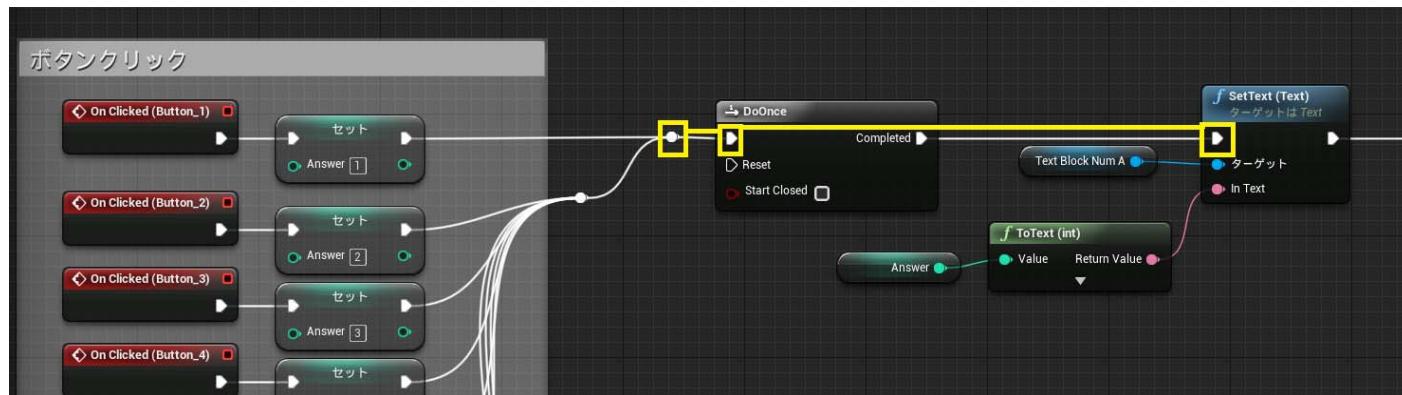
結果表示中にボタンクリックで
結果が変えられる

正解を判定する前にDoOnceノードを追加



入力後にAnswerをセットした後にDoOnceを追加する
右クリック > do onceで検索 > [DoOnce]を選択

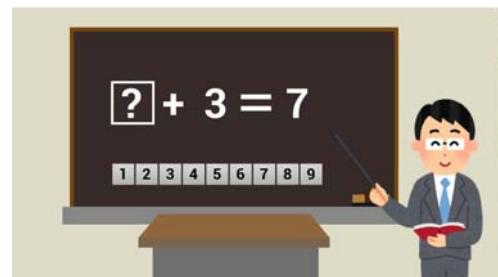
DoOnceを実行するように修正



ボタンをクリック

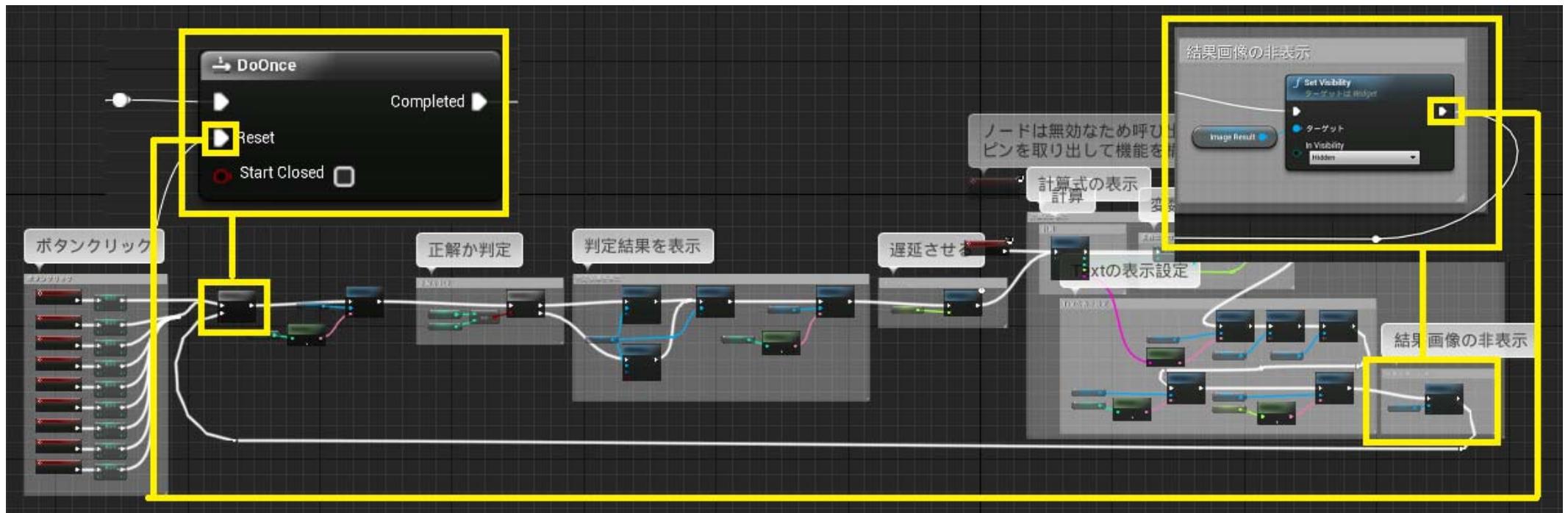


結果がしばらく表示される
結果表示中にボタンをクリックしても結果が変わらない

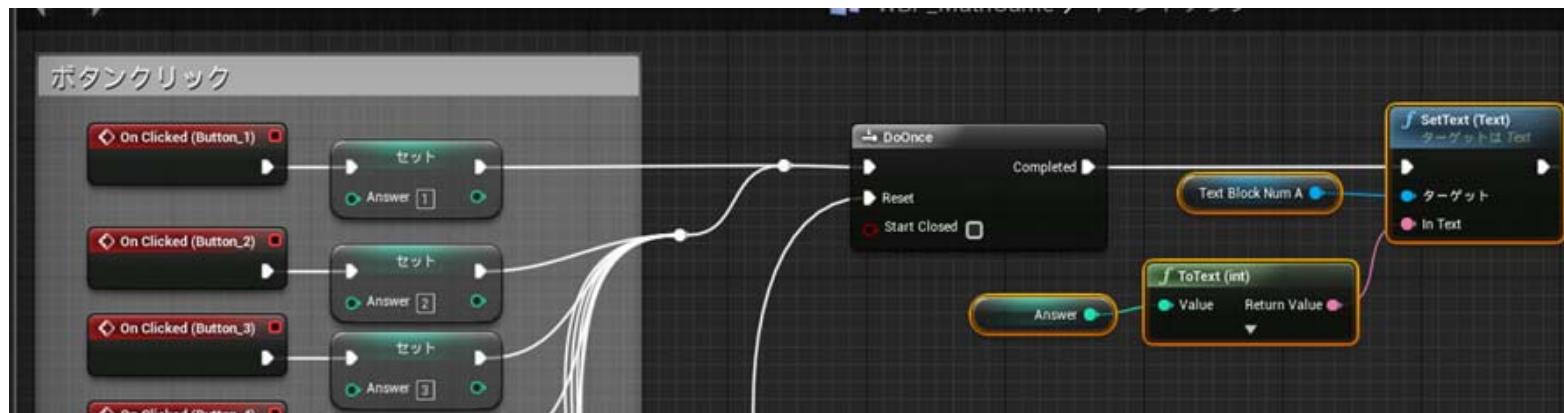


計算式の表示
ボタンを入力しても結果が変わらない

計算式の表示処理後にDoOnceのResetを呼び出す



DoOnceのResetを実行することで 結果表示を行うことが出来る



ボタンをクリック

結果表示
結果表示中にボタンを
クリックしても結果が
変わらない

計算式の表示
ボタンを入力すると結果が表示される

8. テスト項目の作成、デバッグ

8. テスト項目の作成、デバッグ

8.1 そのゲームちゃんと動くの？

8.2 ゲーム内容からテスト項目の抽出

8.3 テスト項目票を作成して、テストを実施

8.4 ブループリント、関数単位のテストするためのデバッグ方法について

8.5 ブラックボックステスト、ホワイトボックステスト

8.6 作りたいゲームの作り方、工程に対応するテスト

8. テスト項目の作成、デバッグ

8.1 そのゲームちゃんと動くの？

8.2 ゲーム内容からテスト項目の抽出

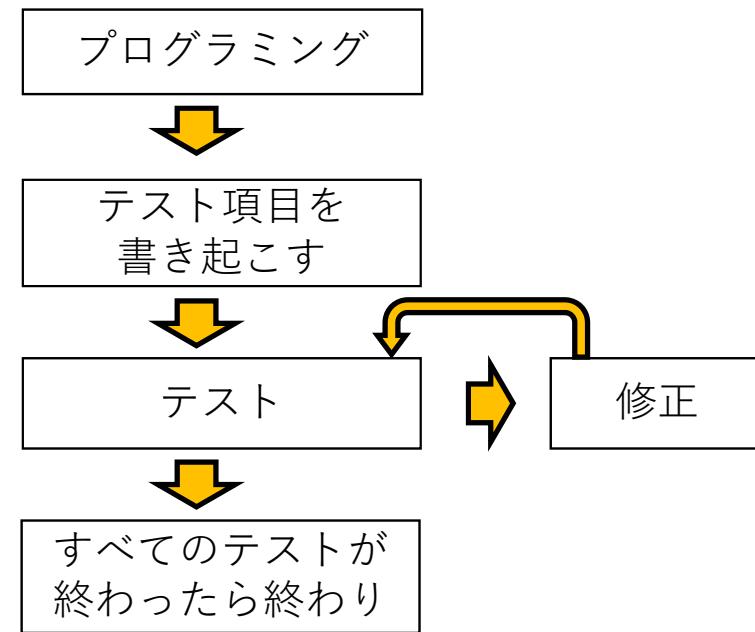
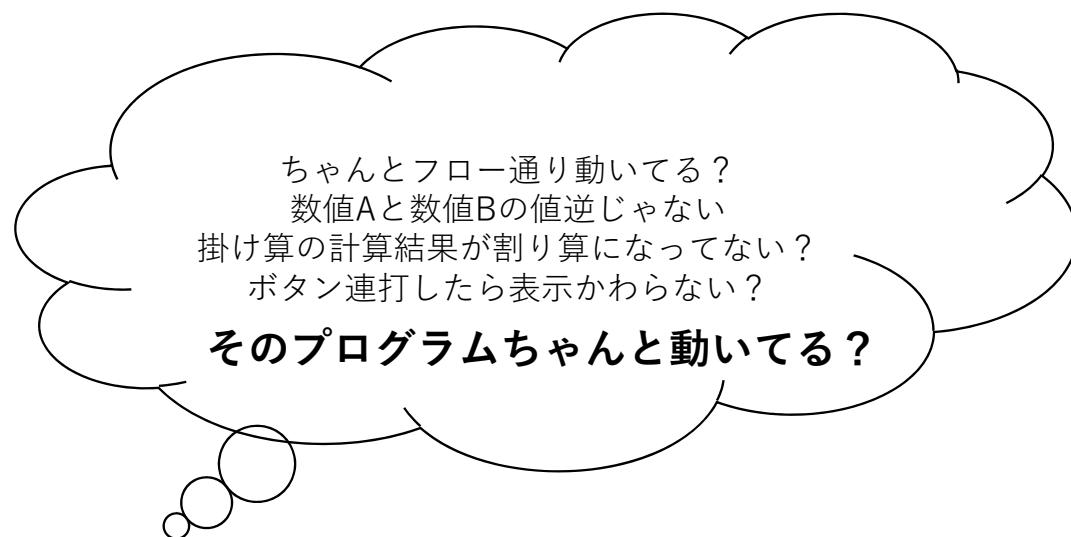
8.3 テスト項目票を作成して、テストを実施

8.4 ブループリント、関数単位のテストするためのデバッグ方法について

8.5 ブラックボックステスト、ホワイトボックステスト

8.6 作りたいゲームの作り方、工程に対応するテスト

作って動いたから終わりではなく、
考えられるすべての動作を確認し終わったら終わり



8. テスト項目の作成、デバッグ

8.1 そのゲームちゃんと動くの？

8.2 ゲーム内容からテスト項目の抽出

8.3 テスト項目票を作成して、テストを実施

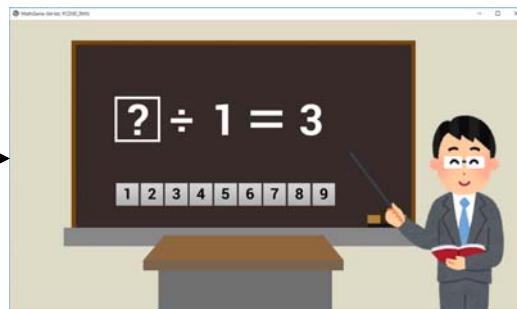
8.4 ブループリント、関数単位のテストするためのデバッグ方法について

8.5 ブラックボックステスト、ホワイトボックステスト

8.6 作りたいゲームの作り方、工程に対応するテスト

作成するゲーム内容からテスト内容を抽出する

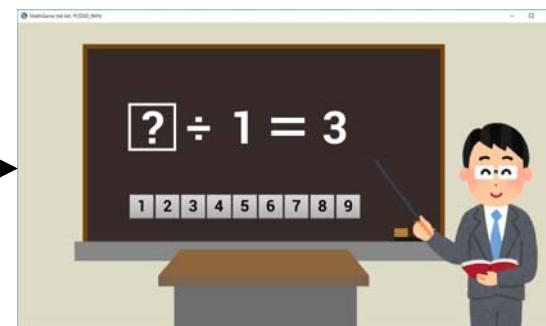
問題出力



数値A、数値Bの数字、
計算方法(+,-,x,÷)がランダム

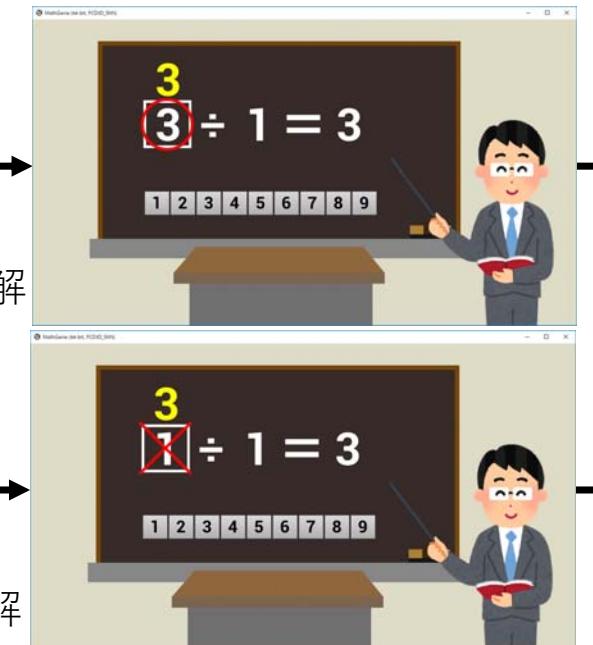
計算式の数値Aは?で
表示されていて分からぬ

入力



数値Aの数字と一致する
ボタン(1~9)をクリック

解答出力

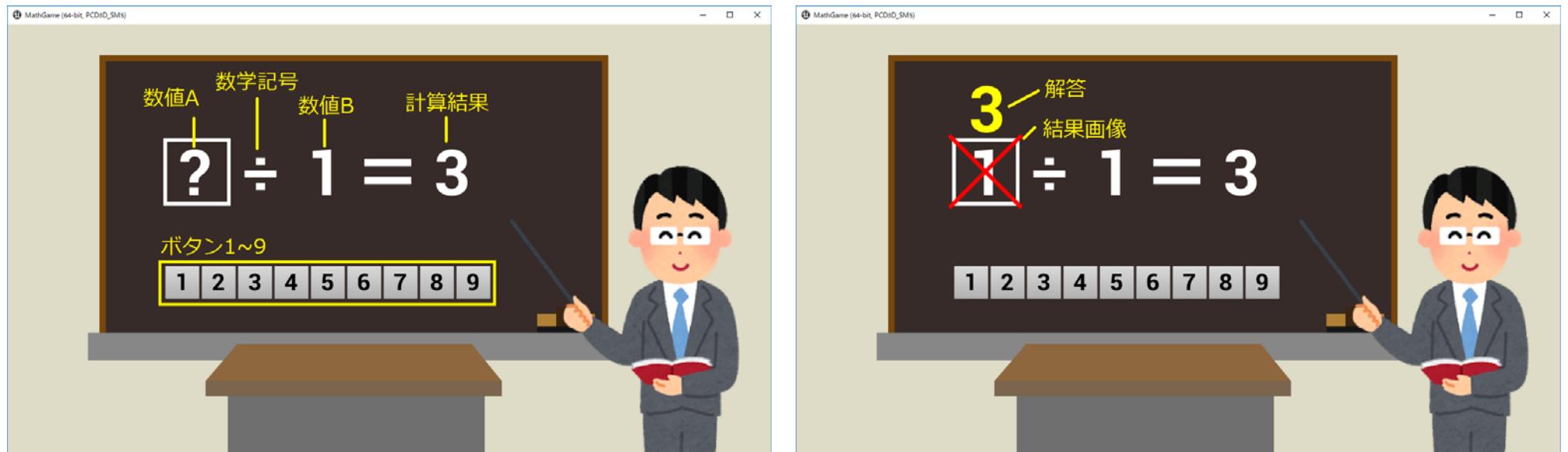


数値Aの値と解答に表示されるランダムで出した
数値が一致するかどうかで正解か不正解か分かる

ESCキー：ゲーム終了

繰り返し

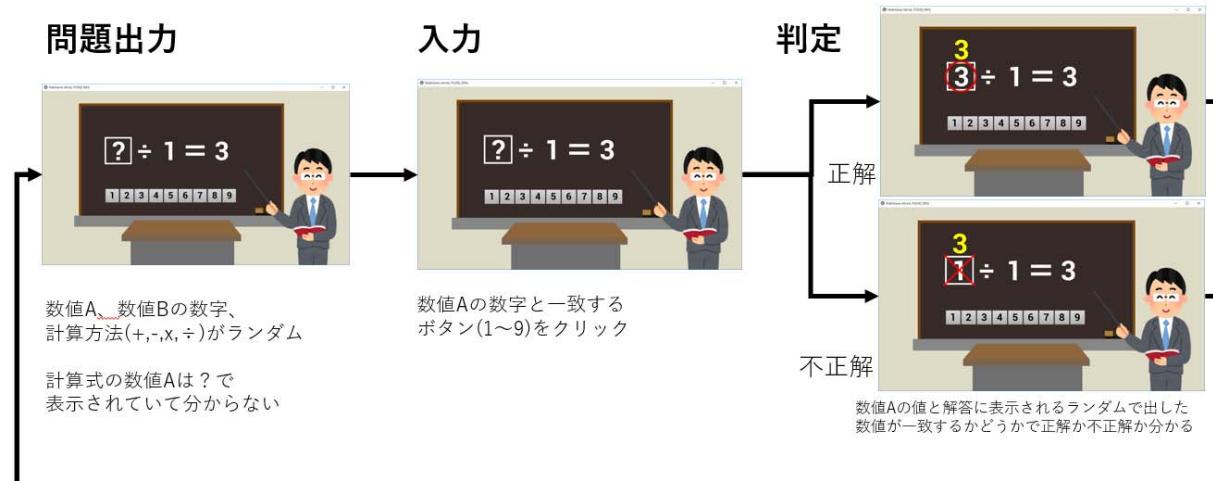
あいまいになっている名称を明記する



テストする項目の名称をあいまいにすると、テスト項目票どおりテストしたけどエラーになる
テスト項目票を起こす前に必ず名称を決めて、テスト項目票を作成する

テスト対象 1

フロー（開始から終了まで）がどうやって動くのか



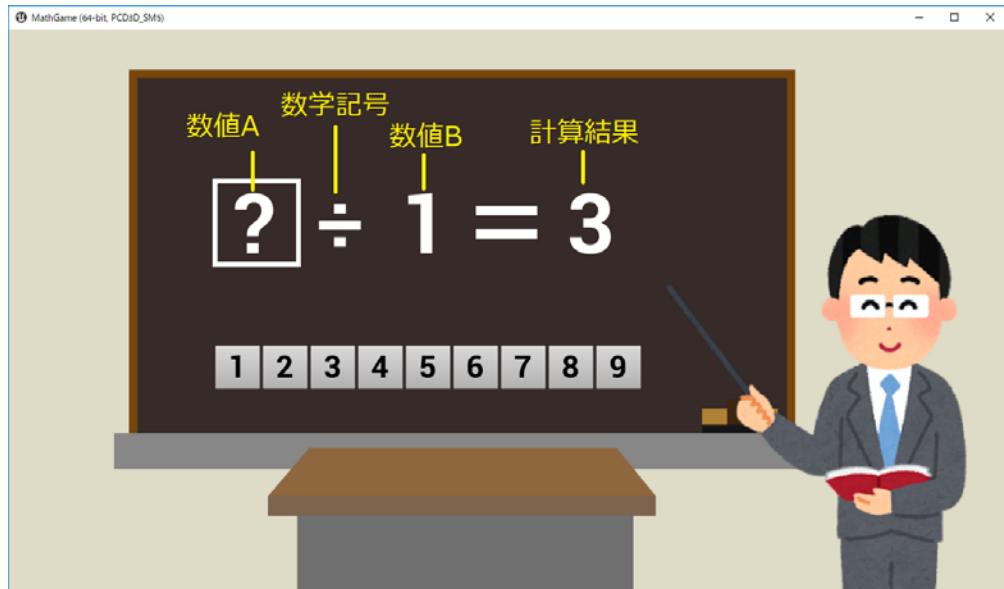
ESCキー：ゲーム終了

繰り返し

開始したら、どのように動いて、どうやったら終了するか

開始	ゲーム開始した時にどうなっているか
終了	どうやったらゲーム終了するか
場面転換のトリガー	何をきっかけに場面が転換するか

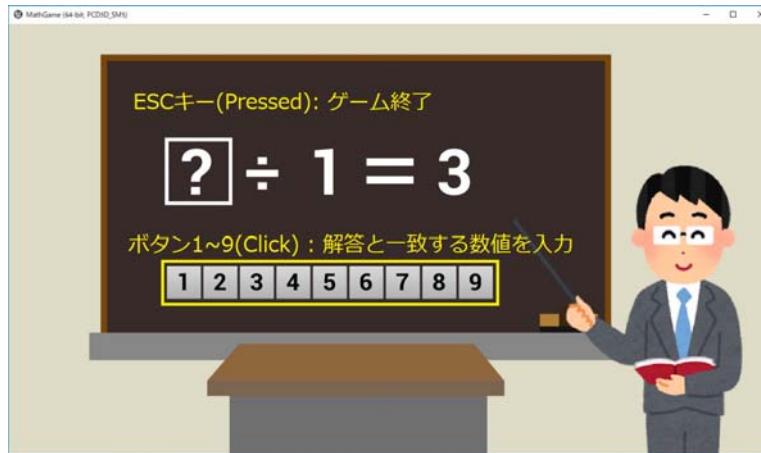
テスト対象 2 表示が変わる



表示が変わるもののは全てテスト対象

テキストが変わる	テキストには何が表示されるのか
表示/非表示	どの場面の時に何が表示されて、何が表示されていないか
画像の切り替え	どの場面の時に何の画像が表示されているのか

テスト対象 3 ユーザー入力



ユーザーが入力するものは全て入力して動作を確認する

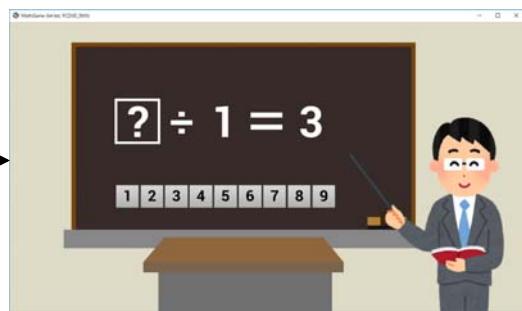
ボタン	似たのような動きをしても、全て確認する Click,Press,Release,Hover,Unhoverなどのイベントを明記
キーボード	何のキーを押した時(Pressed),離した時(Released)を明記
連打対応	必ず連打するユーザーがいるので、連打に対応できるか確認

抽出したテスト項目

- 表示に関連するテスト
- 機能に関連するテスト

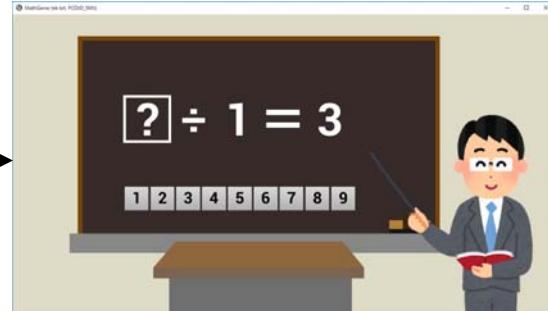
問題出力

- ゲーム開始
- 繰り返し



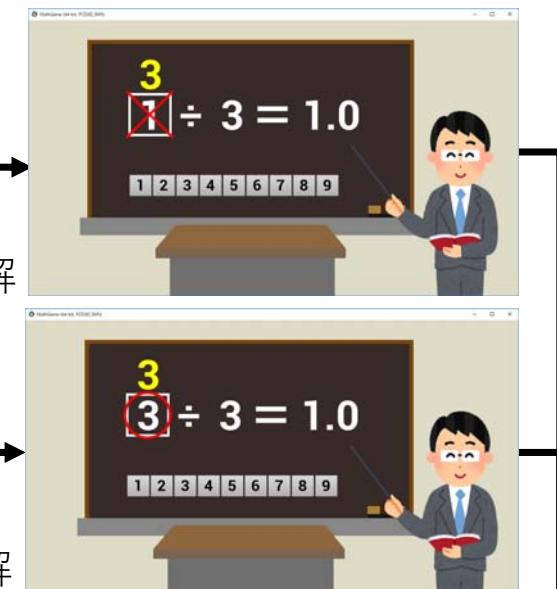
- 数値A、数値Bの数字、計算方法(+,-,x,÷)がランダムで切り替わる
- x4 ○ 計算記号、数値B、合計を表示
計算式の数値Aは?で表示
計算方法4種類テスト
- 結果画像は非表示
- 解答は非表示

入力



- x9 ○ 数値Aの数字と一致するボタン(1~9)をクリックすると解答出力
ボタン全部テスト
- x9 ○ 解答出力中に、ボタンクリックしても解答出力の表示が変わらない
連打対応

解答出力



正解

不正解

- x4 ○ 表示されている計算結果が正しい
数値A: ボタンの入力値を表示
解答 : 数値Aのランダム数字を表示
正解、不正解で表示する画像を変える
- x9 ○ 正解 : 数値Aと解答が一致
不正解 : 数値Aと解答が不一致
- x2 ○

○ ESCキー：ゲーム終了

○ 繰り返し

8. テスト項目の作成、デバッグ

8.1 そのゲームちゃんと動くの？

8.2 ゲーム内容からテスト項目の抽出

8.3 テスト項目票を作成して、テストを実施

8.4 ブループリント、関数単位のテストするためのデバッグ方法について

8.5 ブラックボックステスト、ホワイトボックステスト

8.6 作りたいゲームの作り方、工程に対応するテスト

テスト項目票

項目番号	大項目	中項目	小項目	テスト内容	結果
1-1	問題出力	数値A		? が表示される	
1-2-1		数学記号	足し算(+)	+ が表示される	
1-2-2			引き算(-)	- が表示される	
1-2-3			掛け算(×)	× が表示される	
1-2-4			割り算(÷)	÷ が表示される	
1-3		数値B		繰り返し毎にランダムの数値が表示される	
1-4		計算結果		繰り返し毎にランダムの数値が表示される	
1-5		解答		非表示	
1-6		結果画像		非表示	

項目番号	大中小項目を数値化してIDにした番号（エラーを報告する際に使う）
大・中・小項目	テスト項目のカテゴリ分け（3or5くらいの階層で終わるように）
テスト内容	何がどうなれば正しいのか記述する
結果	正しかったら○を記入する

算数ゲームでは作成したテスト項目票を使ってテスト

決まったフォーマットはない。エクセルで作られることが多い。

規模が大きくなると、実施した人の名前、消化した日付、エラーを報告する報告書があるときは報告書のIDなど記入しなくてはならない項目が増える
エラーを発見出来るテストが書かれているテスト項目票がいいテスト項目票

テスト項目票(表示：問題出力)

テスト項目票(表示：解答出力)

項目番号	大項目	中項目	小項目	テスト内容	結果
2-1-1	解答出力	数値A	ボタン1(Click)	1が表示される	
2-1-2			ボタン2(Click)	2が表示される	
2-1-3			ボタン3(Click)	3が表示される	
2-1-4			ボタン4(Click)	4が表示される	
2-1-5			ボタン5(Click)	5が表示される	
2-1-6			ボタン6(Click)	6が表示される	
2-1-7			ボタン7(Click)	7が表示される	
2-1-8			ボタン8(Click)	8が表示される	
2-1-9			ボタン9(Click)	9が表示される	
2-2		数学記号		問題出力から変化なし	
2-3		数値B		問題出力から変化なし	
2-4		計算結果		問題出力から変化なし	
2-5		解答		繰り返し毎にランダムの数値が表示される	
2-6-1		結果画像	数値Aと解答が一致	○の画像が表示される	
2-6-2			数値Aと解答が不一致	×の画像が表示される	

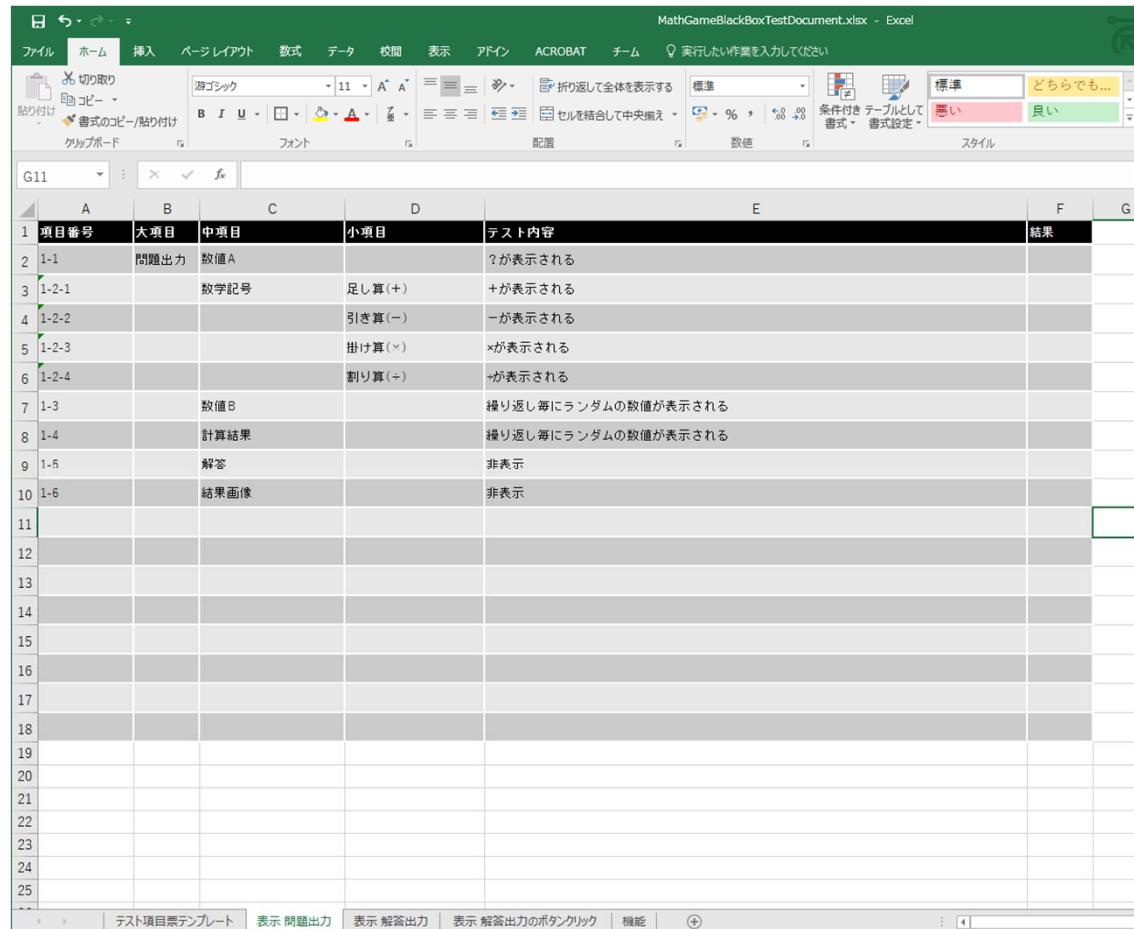
テスト項目票

(表示：解答出力のボタンクリック)

テスト項目票(機能)

項目番号	大項目	中項目	小項目	テスト内容	結果
4-1	フロー	開始		問題出力されること	
4-2-1		問題出力中の ボタンクリック	ボタン1(Click)	解答出力されること	
4-2-2			ボタン2(Click)	解答出力されること	
4-2-3			ボタン3(Click)	解答出力されること	
4-2-4			ボタン4(Click)	解答出力されること	
4-2-5			ボタン5(Click)	解答出力されること	
4-2-6			ボタン6(Click)	解答出力されること	
4-2-7			ボタン7(Click)	解答出力されること	
4-2-8			ボタン8(Click)	解答出力されること	
4-2-9			ボタン9(Click)	解答出力されること	
4-3		解答出力 > 問題出力		解答出力から自動的に問題出力されること	
4-4		終了	ESCキー(Pressed)	ゲームが終了されること	
4-5-1		計算結果	足し算(+)	解答出力の計算が正しいこと(解答+数値B=計算結果)	
4-5-2			引き算(-)	解答出力の計算が正しいこと(解答 - 数値B=計算結果)	
4-5-3			掛け算(×)	解答出力の計算が正しいこと(解答 × 数値B=計算結果)	
4-5-4			割り算(÷)	解答出力の計算が正しいこと(解答 ÷ 数値B=計算結果)	

Excelで作成することが多い



The screenshot shows an Excel spreadsheet titled "MathGameBlackBoxTestDocument.xlsx". The ribbon menu is visible at the top. The main table has columns labeled A through G. Column A is "項目番号", B is "大項目", C is "中項目", D is "小項目", E is "テスト内容", F is "結果", and G is empty. Rows 1 through 10 contain specific test cases, and rows 11 through 25 are empty. Row 10 is highlighted in light blue. The status bar at the bottom shows tabs for "テスト項目票テンプレート", "表示 問題出力" (selected), "表示 解答出力", "表示 解答出力のボタンクリック", "機能", and a plus sign.

項目番号	大項目	中項目	小項目	テスト内容	結果	
1						
2	1-1	問題出力	数値A	?が表示される		
3	1-2-1	数学記号	足し算(+)	+が表示される		
4	1-2-2		引き算(-)	-が表示される		
5	1-2-3		掛け算(×)	*が表示される		
6	1-2-4		割り算(÷)	÷が表示される		
7	1-3	数値B		繰り返し毎にランダムの数値が表示される		
8	1-4	計算結果		繰り返し毎にランダムの数値が表示される		
9	1-5	解答		非表示		
10	1-6	結果画像		非表示		
11						
12						
13						
14						
15						
16						
17						
18						
19						
20						
21						
22						
23						
24						
25						

プレゼン資料にまとめたかったので、
プレゼン資料にテスト項目票を載せましたが、
Excelでテスト項目票を作成することが多い

ExcelやPowerPointを持っていない人は Google ドライブにアップすることで編集可能

ダウンロードしたファイル（Excel, PowerPoint）を
ドラッグ & ドロップ

ExcelがPCにインストールされていなくても
Googleスプレッドシートで編集可能

Google スpreadsheet screenshot showing the document structure:

項目番号	A	B	C	D	E	F
1	項目番号	大項目	中項目	小項目	テスト内容	結果
2	1-1	問題出力	数値A		?が表示される	
3	1-2-1	数学記号	足し算(+)	+が表示される		
4	1-2-2		引き算(-)	-が表示される		
5	1-2-3		掛け算(*)	*が表示される		
6	1-2-4		割り算(÷)	÷が表示される		
7	1-3	数値B			繰り返し毎にランダムの数値が表示される	
8	1-4	計算結果			繰り返し毎にランダムの数値が表示される	
9	1-5	解答			非表示	
10	1-6	結果画像			非表示	

Google スpreadsheet screenshot showing the document structure:

	A	B	C	D	E	F
1	項目番号	大項目	中項目	小項目	テスト内容	結果
2	1-1	問題出力	数値A		?が表示される	
3	1-2-1	数学記号	足し算(+)	+が表示される		
4	1-2-2		引き算(-)	-が表示される		
5	1-2-3		掛け算(*)	*が表示される		
6	1-2-4		割り算(÷)	÷が表示される		
7	1-3	数値B			繰り返し毎にランダムの数値が表示される	
8	1-4	計算結果			繰り返し毎にランダムの数値が表示される	
9	1-5	解答			非表示	
10	1-6	結果画像			非表示	
11						
12						

8. テスト項目の作成、デバッグ

8.1 そのゲームちゃんと動くの？

8.2 ゲーム内容からテスト項目の抽出

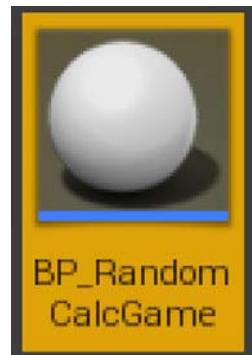
8.3 テスト項目票を作成して、テストを実施

8.4 ブループリント、関数単位のテストするためのデバッグ方法について

8.5 ブラックボックステスト、ホワイトボックステスト

8.6 作りたいゲームの作り方、工程に対応するテスト

ブループリントの設計書がある場合



ブループリントクラス名	親クラス	機能内容
BP_RandomCalcGame	Actor	<p>ランダムの数値A,Bをランダムの計算方法(足し算、引き算、掛け算、割り算)で計算した結果を計算式にして出力する。計算式の数値Aは?で出力される。</p> <p>キーボード1~9を押す(Pressed)とランダムで算出した数値Aと比較を行う</p> <p>キーボード入力の数値とランダムで算出した数値Aを比較</p> <p>数値が一致していれば、Trueを出力</p> <p>数値が一致していないければ、Falseを出力</p> <p>結果を出力した後、再び計算式を出力する</p>

関数名	処理内容	I/O	パラメータ名	型
Plus	引数A,Bを足した結果をTotalで返す	I	A	Integer
		I	B	Integer
		O	Total	Integer

関数名	処理内容	I/O	パラメータ名	型
RandomCalculation	ランダムの数値A,Bを算出 ランダムで計算方法を切り替える(足し算、引き算、掛け算、割り算) ランダムな数値A,Bとランダムな計算方法の 計算結果をTotalに設定し、計算した数学記号 をCalcTypeに設定する	O	A	Integer
		O	B	Integer
		O	Total	Integer
		O	CalcType	String

I:インプット O:アウトプットの略

BP_RandomCalcGameをテストする



BP_RandomCalcGameを実行するレベルである
RandomCalcGameを開く

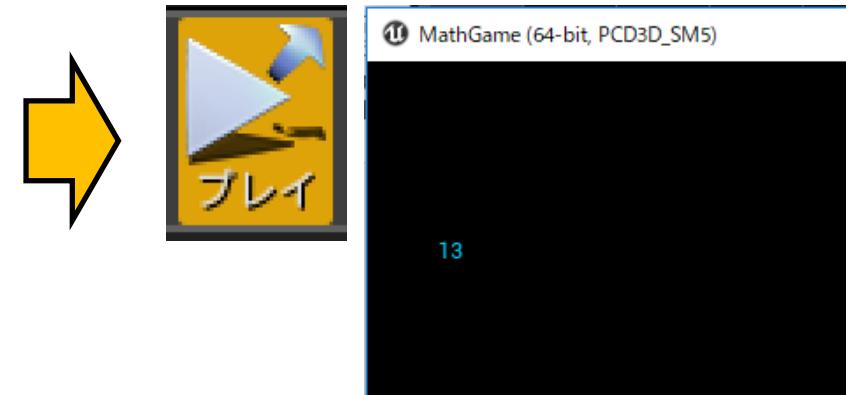
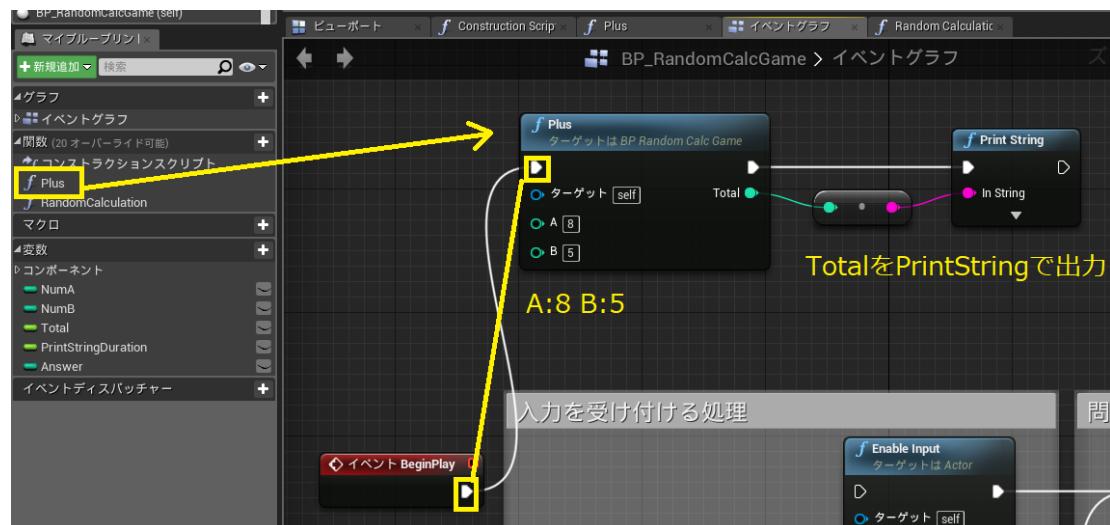


BP_RandomCalcGameを開く

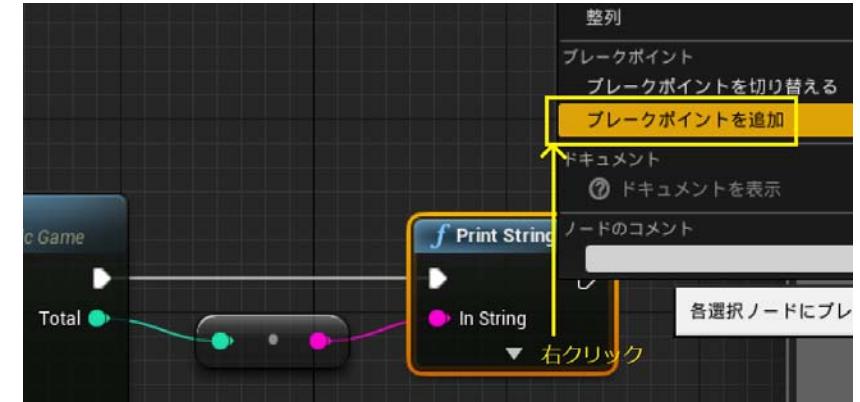
関数のテスト方法 (print デバッグ)

関数名	処理内容	I/O	パラメータ名	型	値	結果
Plus	引数A,Bを足した結果をTotalで返す	I	A	Integer	8	
		I	B	Integer	5	
		O	Total	Integer	13	

設計書の右側に値、結果を追加
インプットの値を決めて、正解であるアウトプットの値を書く

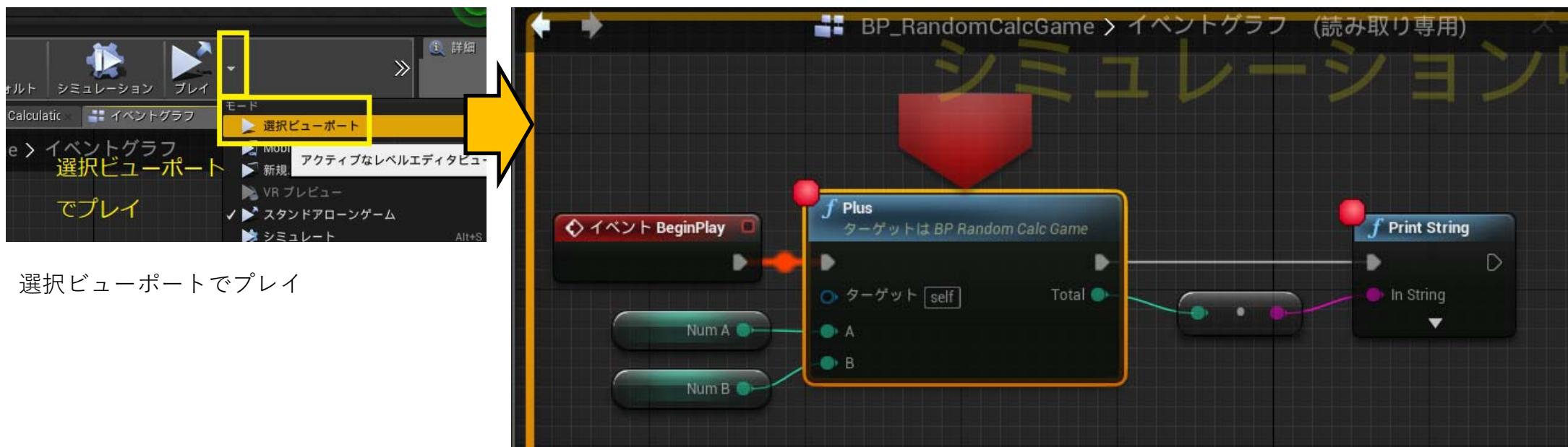


関数のテスト方法 (ブレークポイント、ウォッチ デバッグ)



ノードの左上に赤い●が表示される

ブレークポイントで処理が止まる



ブレークポイントで処理が止まる
処理が止まる = ノードがつながっている と判断することが出来る
処理を書いたはずなのに動いていないときは、ブレークポイントを追加して確認する

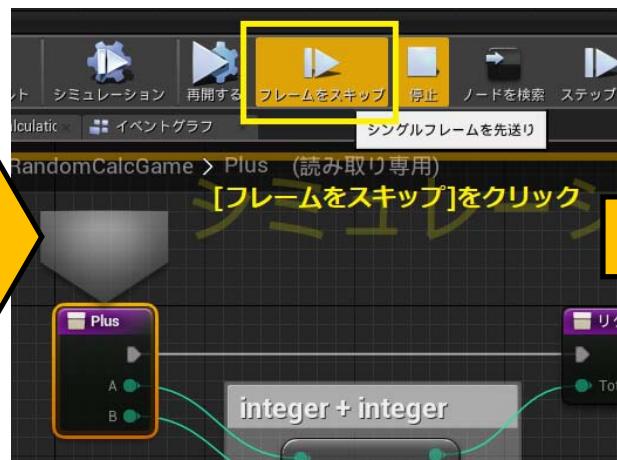
フレームをスキップ[°]



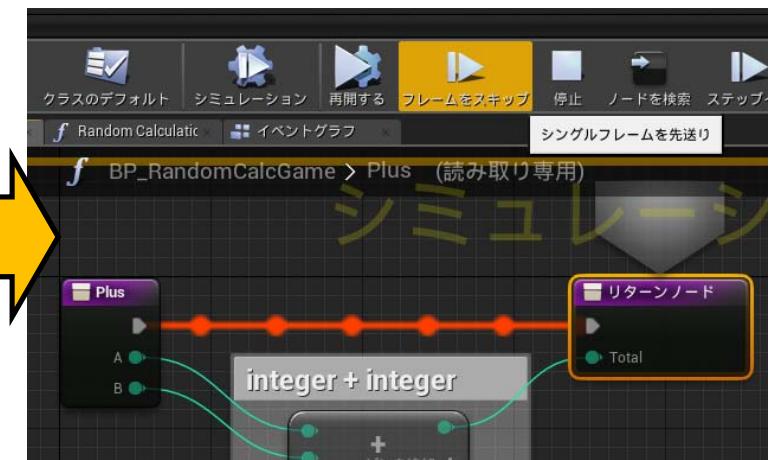
次に処理をするノードに移動する



[フレームをスキップ]をクリック



次に処理するノードである
関数Plusの最初のノードに移動する



次に処理するノードである
リターンノードに移動する

もう一度[フレームスキップ]をクリック

再開する



処理を再開する

次のブレークポイントがあれば、次のブレークポイントに移動
次のブレークポイントがなければ、ゲームが再開される

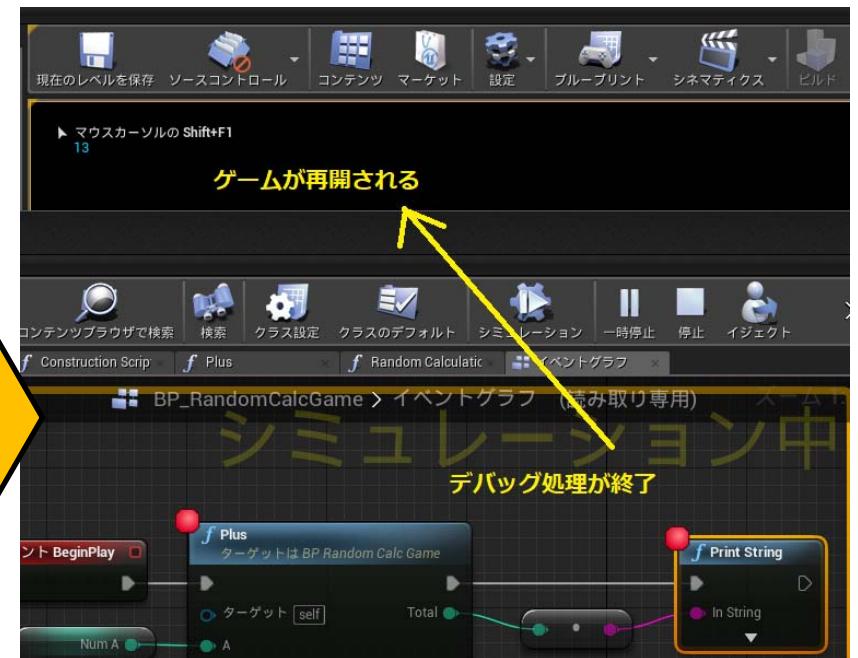


[再開する]をクリック



次のブレークポイントがある場合は
次のブレークポイントに移動する

もう1度、[再開する]をクリック

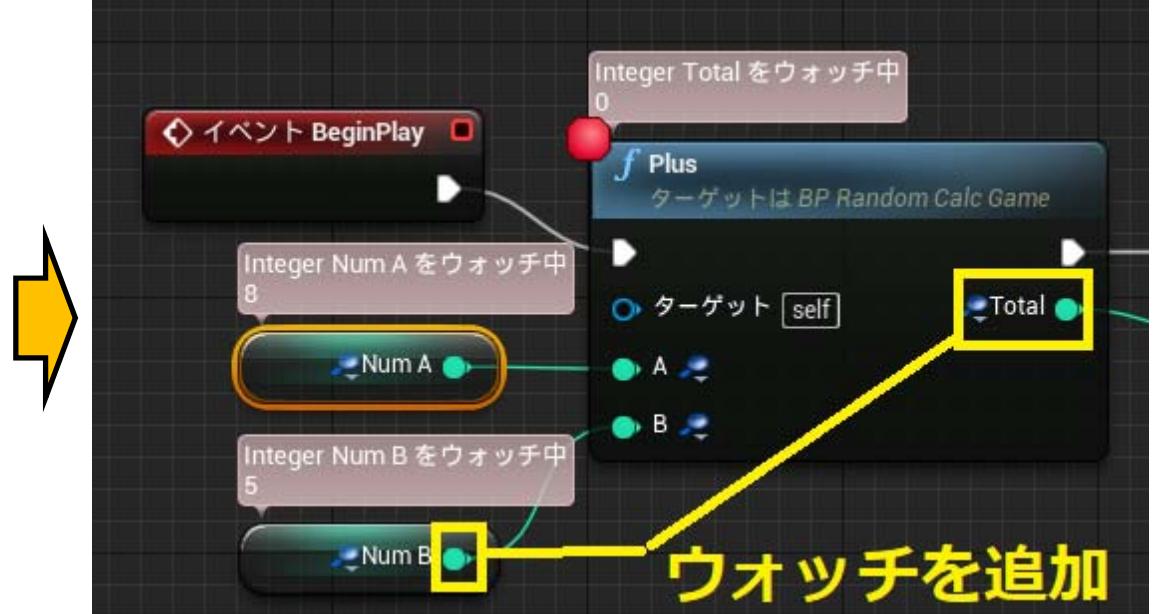


次のブレークポイントがない場合は
ゲームが再開されるに移動する

ウォッチを追加



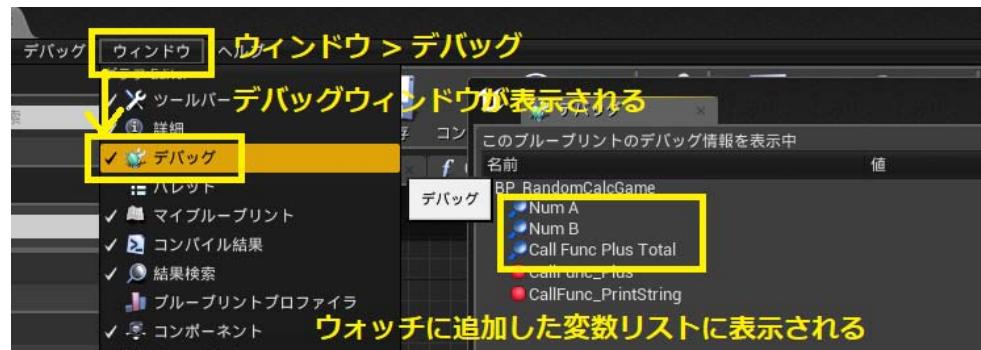
変数NumAのゲットを右クリック > [この値をウォッチ]を選択



ウォッチを追加すると吹き出しが表示され値が表示される
変数NumB,関数PlusのアウトプットTotalにウォッチを追加する

デバッグウィンドウでウォッチの値を確認

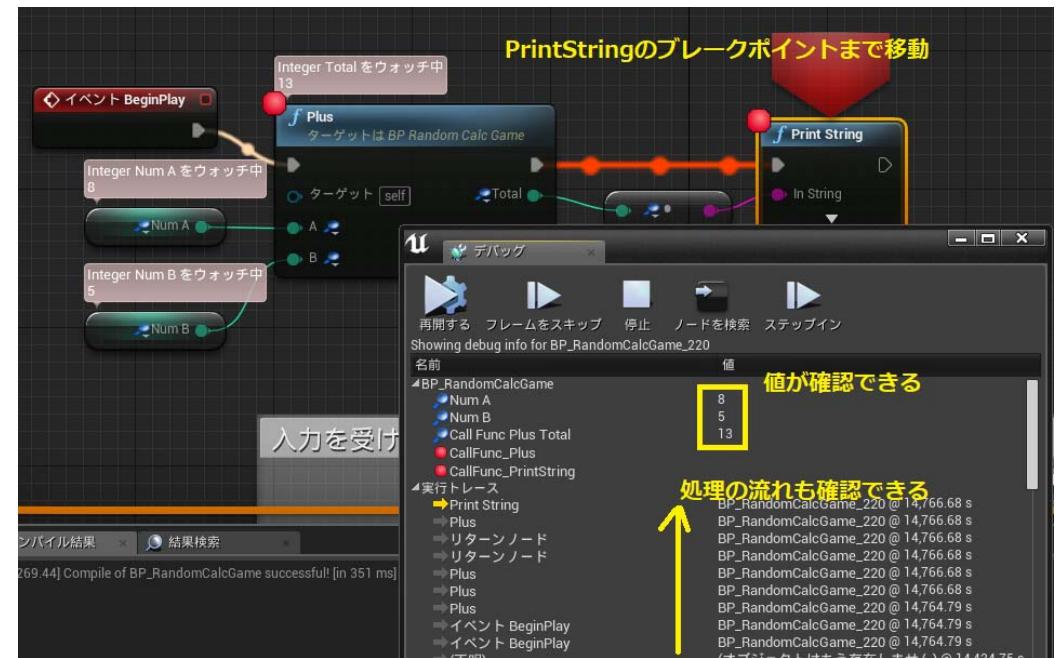
関数名	処理内容	I/O	パラメータ名	型	値	結果
Plus	引数A,Bを足した結果をTotalで返す	I	A	Integer	8	
		I	B	Integer	5	
		O	Total	Integer	13	



ウィンドウ > デバッグ
デバッグウィンドウが表示される
ウォッチに追加した変数がリストに追加される

PrintStringのブレークポイントまで移動する
デバッグウィンドウで変数の値を確認する

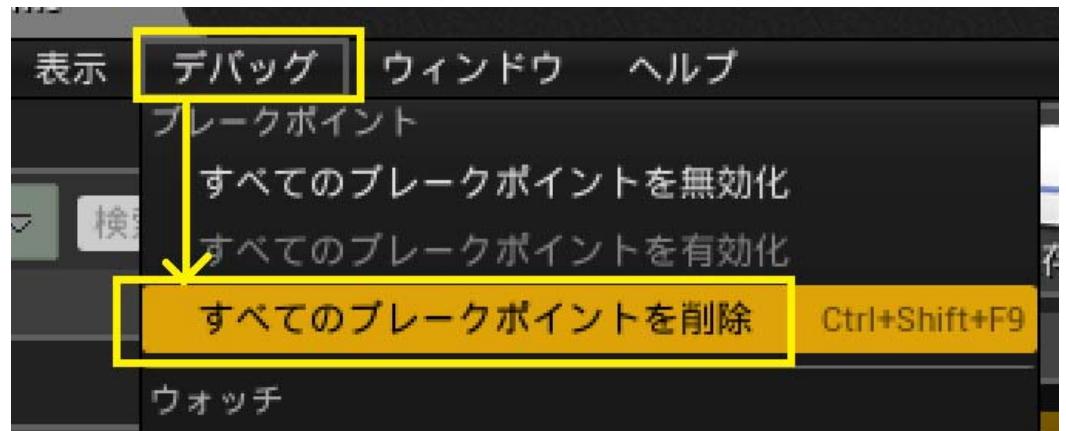
実行トレースを見ると処理の流れも確認できる



ブレークポイントの削除



ブレークポイントを追加したノードを右クリック
> [ブレークポイントを取り除く]を選択

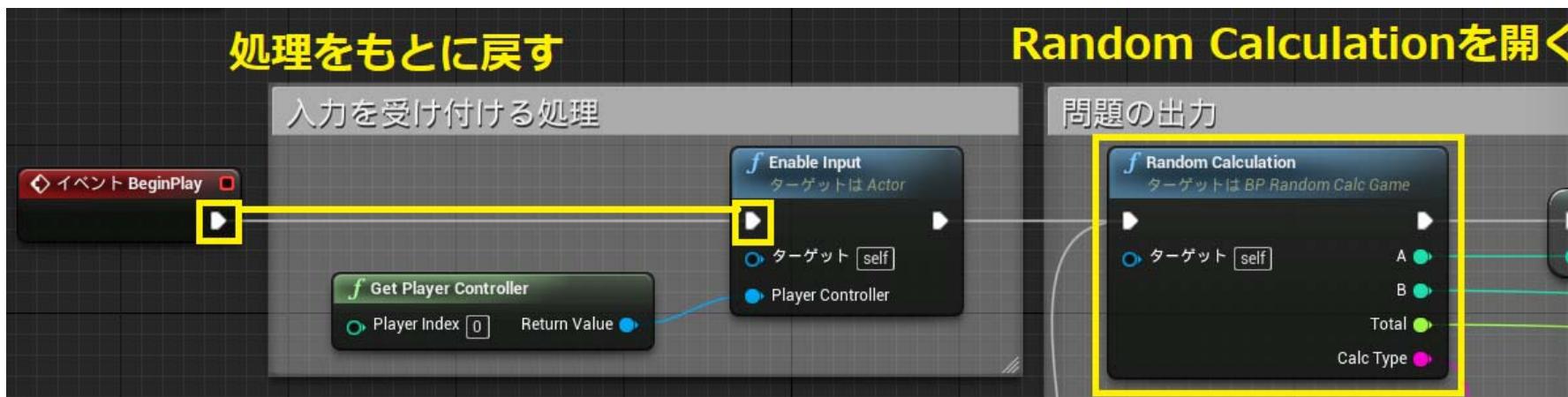


一括でブレークポイントを削除
デバッグ > [すべてのブレークポイントを削除]を選択

アウトプットしかない RandomCalculationのテスト項目

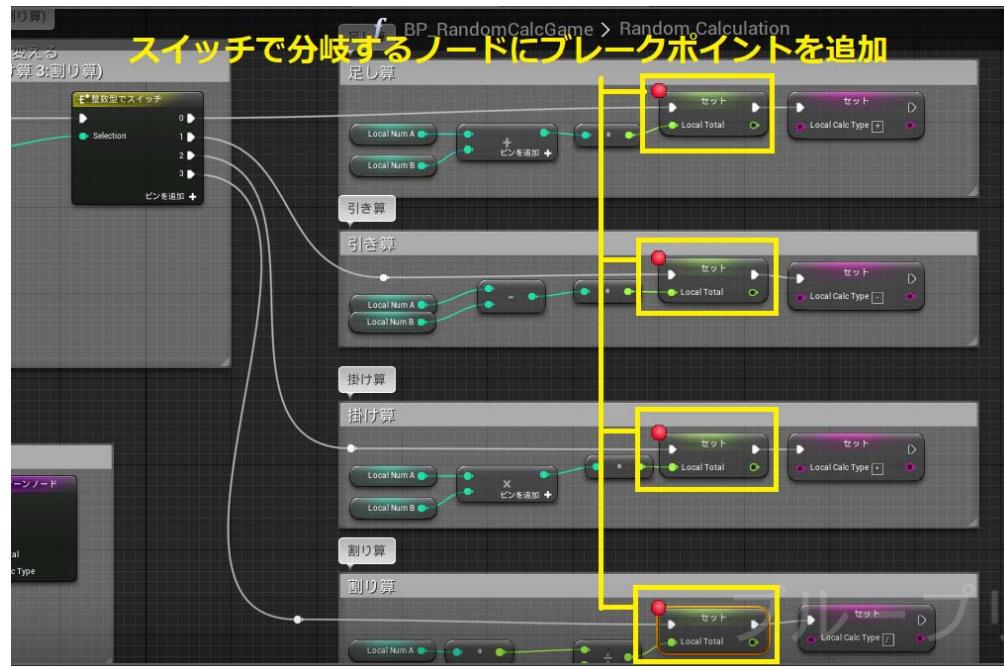
関数名	処理内容	I/O	パラメータ名	型	結果
RandomCalculation	ランダムの数値A,Bを算出 ランダムで計算方法を切り替える(足し算、引き算、掛け算、割り算) ランダムな数値A,Bとランダムな計算方法の計算結果をTotalに設定し、計算した数学記号をCalcTypeに設定する	O	A	Integer	ランダムの値
		O	B	Integer	ランダムの値
		O	Total	Integer	足し算(A + B=Total)
					引き算(A - B=Total)
					掛け算(A × B=Total)
					割り算(A ÷ B=Total)
		O	CalcType	String	足し算 : +
					引き算 : -
					掛け算 : *
					割り算 : /

RandomCalculationを開く

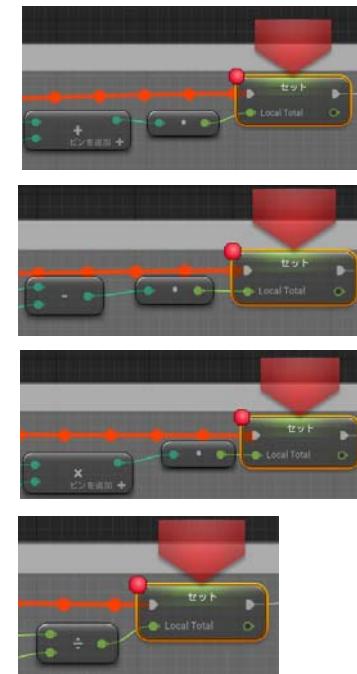
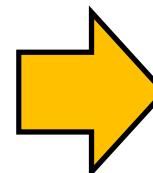


全パスチェック

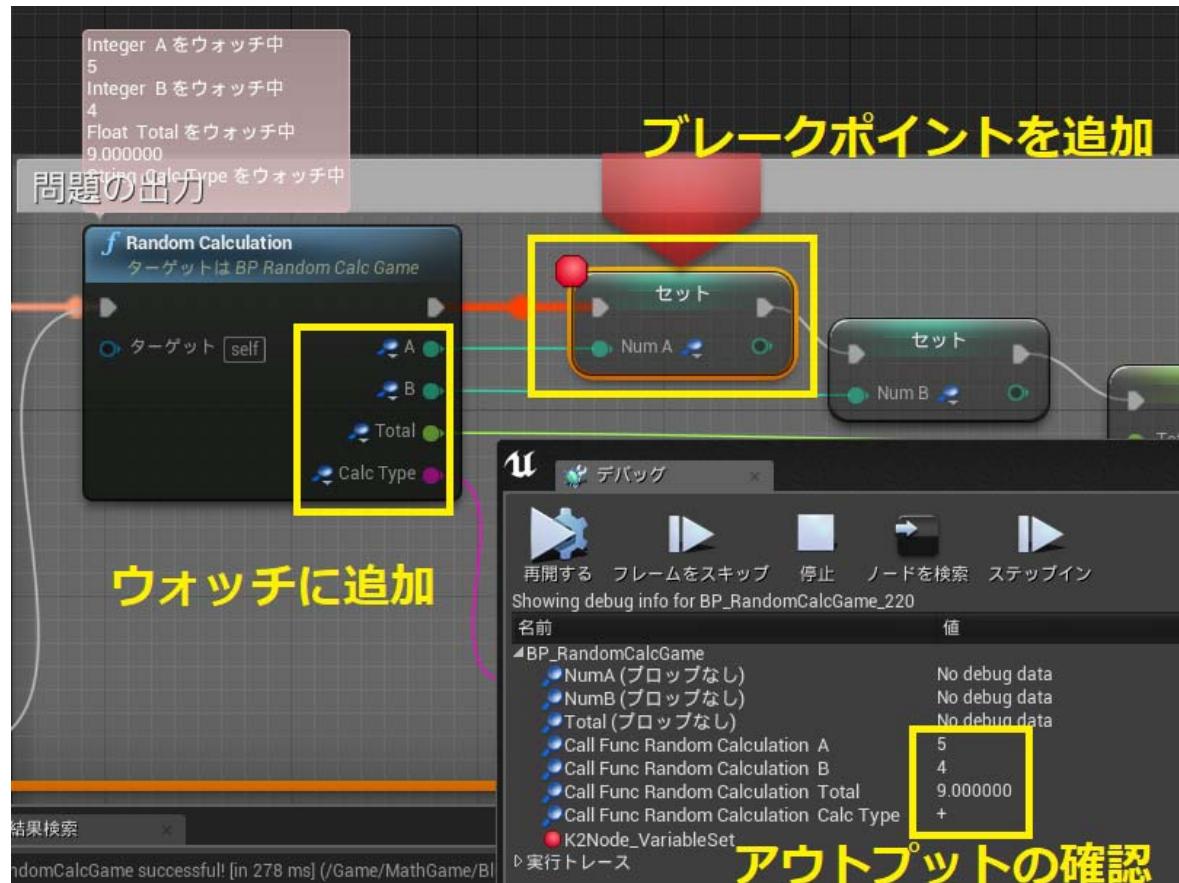
関数内の分岐 (Branch, Switch) をすべて通過することを確認する



インプット、アウトプットの確認以外にも
関数内で分岐がある場合はすべての処理を通過することを確認する



RandomCalculationのアウトプットを確認

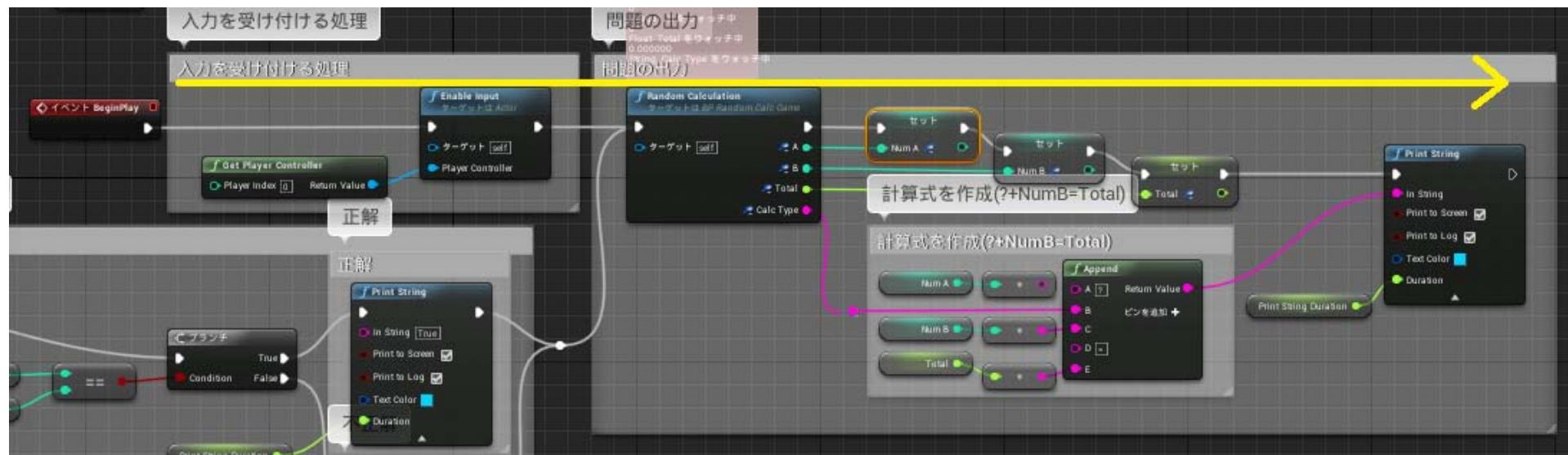


RandomCalculation後のノードに
ブレークポイントを追加

RandomCalculationをウォッチに追加

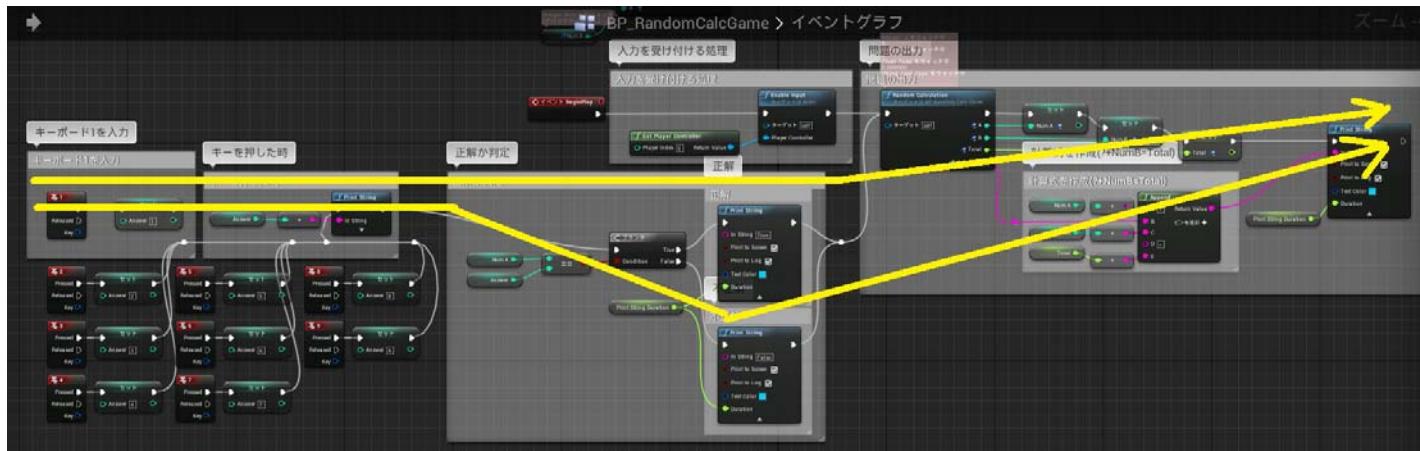
ブレークポイントまで移動させて、
デバッグウィンドウでアウトプットを確認する

ブループリントの全パスチェック 開始時のパスをチェック



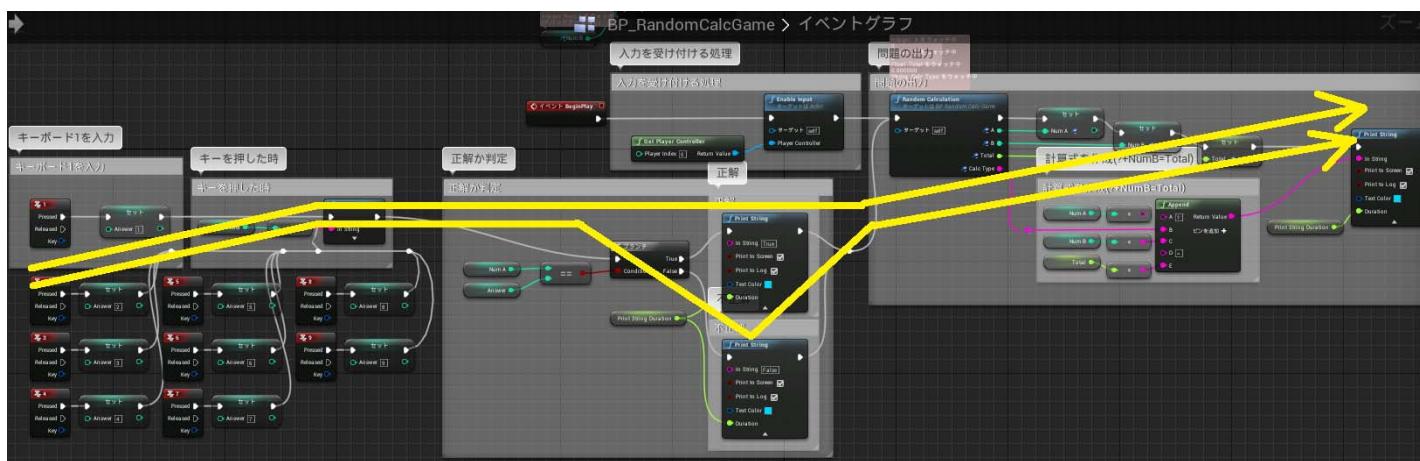
BeginPlayから計算式の出力まで1パスとして確認

キーボード入力の全パスチェック



キーボード1入力 > 判定
> 計算式出力 までを1パス

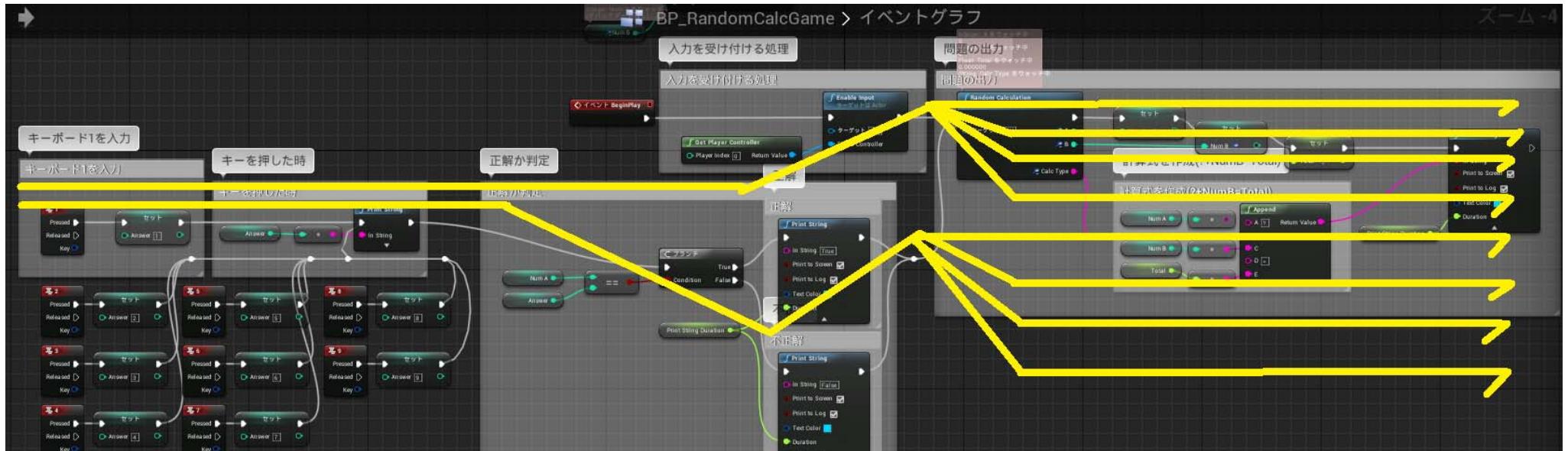
途中ブランチがあるので、
2パスになる



キーボード2入力 > 判定
> 計算式出力 までを1パス

キーボード2も2パスになる

RandomCalculationを作成しない場合の全パスチェック



関数RandomCalculationを作っていないと、計算式の作成の際にスイッチで分岐することになるので、
全 $2 \times 4 = 8$ パスになる。

キー入力が1～9まであるので、 $8 \times 9 = 72$ パスの全パステストになる
RandomCalculationを作っていることで、 $2 \times 9 = 18$ パスのテストで済む

なぜ関数を作るか。1つの理由としてテスト項目を減らす意味がある。

8. テスト項目の作成、デバッグ

8.1 そのゲームちゃんと動くの？

8.2 ゲーム内容からテスト項目の抽出

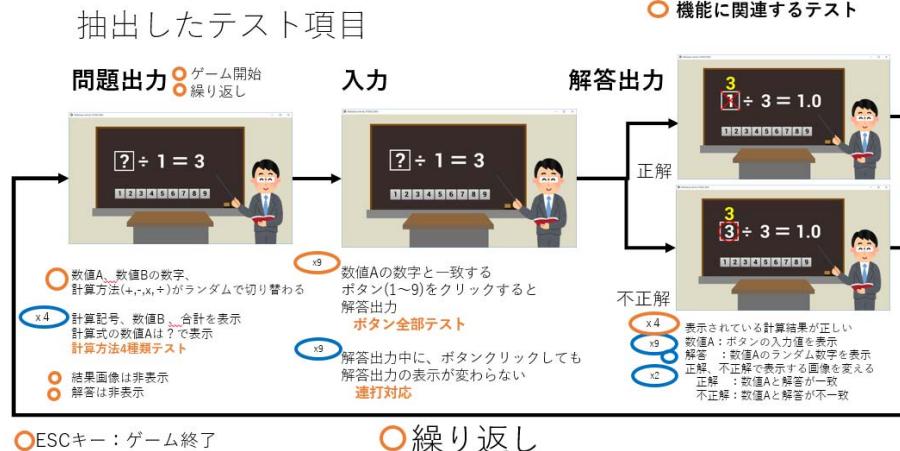
8.3 テスト項目票を作成して、テストを実施

8.4 ブループリント、関数単位のテストするためのデバッグ方法について

8.5 ブラックボックステスト、ホワイトボックステスト

8.6 作りたいゲームの作り方、工程に対応するテスト

ブラックボックステストと ホワイトボックステスト

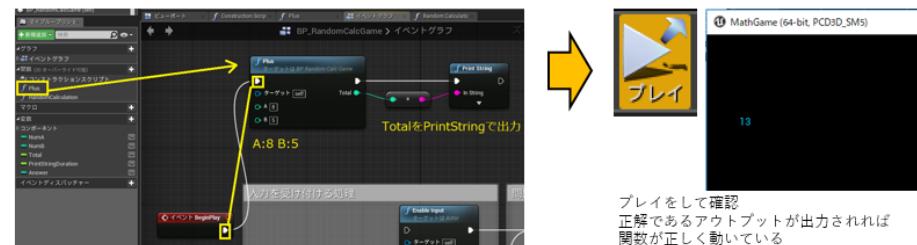


ブラックボックステスト（ユーザー視点）
プログラムの中身はどうなっていてもいいので、
期待した動作をするか確認するテスト

関数のテスト方法 (print デバッグ)

関数名	処理内容	I/O	パラメータ名	型	値	結果
Plus	引数A,Bを足した結果をTotalで返す	I	A	Integer	8	OK
		I	B	Integer	5	
		O	Total	Integer	13	

設計書の右側に値、結果を追加
インプットの値を決めて、正解であるアウトプットの値を書く



プレイをして確認
正解であるアウトプットが表示されれば
関数が正しく動いている

270

ホワイトボックステスト（開発者視点）

プログラムの中身が正しく動いているか確認するテスト
(I/O確認、全パスチェック)

8. テスト項目の作成、デバッグ

8.1 そのゲームちゃんと動くの？

8.2 ゲーム内容からテスト項目の抽出

8.3 テスト項目票を作成して、テストを実施

8.4 ブループリント、関数単位のテストするためのデバッグ方法について

8.5 ブラックボックステスト、ホワイトボックステスト

8.6 作りたいゲームの作り方、工程に対応するテスト

作りたいゲームを作るためにブレイクダウンする

ブループリントを学習する
算数ゲームを作りたい



算数ゲームの中身と
フローを書き起こす



実装しないといけない機能を
詳細に書き起こす



機能ごとの作り方を考える

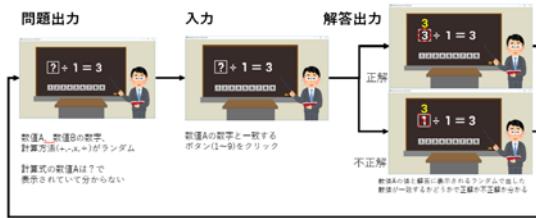


プログラミング

算数ゲーム

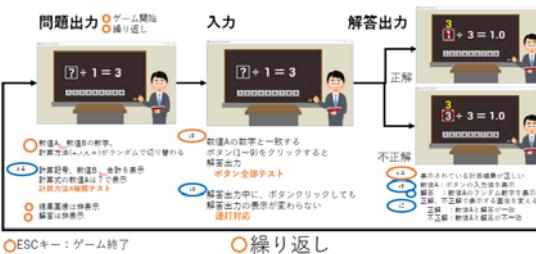
ブループリント

作成するゲーム内容



ESCキー：ゲーム終了 繰り返し

抽出したテスト項目



ESCキー：ゲーム終了 繰り返し

ブループリントの設計書がある場合



工程に対応するテスト項目票を作成する

ブループリントを学習する
算数ゲームを作りたい

算数ゲームの中身と
フローを書き起こす

実装しないといけない機能を
詳細に書き起こす

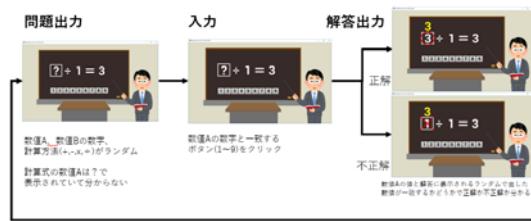
機能ごとの作り方を考える

プログラミング

算数ゲーム

ブループリント

作成するゲーム内容



プレイしてみて
作ろうとしていた
算数ゲームが出来ているか

ESCキー: ゲーム終了 繰り返し

抽出したテスト項目

問題出力

入力

解答出力

ESCキー: ゲーム終了

繰り返し

表示に関連するテスト

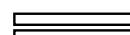
機能に関連するテスト

テスト項目票(表示: 解答出力)

問題番号	出題範囲	中間値	テスト用値	結果
2-1	算数実行	教A	1	正解
2-2	算数実行	教B	2	正解
2-3	算数実行	教C	3	正解
2-4	算数実行	教D	4	正解
2-5	算数実行	教E	5	正解
2-6	算数実行	教F	6	正解
2-7	算数実行	教G	7	正解
2-8	算数実行	教H	8	正解
2-9	算数実行	教I	9	正解
2-10	算数実行	教J	10	正解
2-11	算数実行	教K	11	正解
2-12	算数実行	教L	12	正解
2-13	算数実行	教M	13	正解
2-14	算数実行	教N	14	正解
2-15	算数実行	教O	15	正解
2-16	算数実行	教P	16	正解
2-17	算数実行	教Q	17	正解
2-18	算数実行	教R	18	正解
2-19	算数実行	教S	19	正解
2-20	算数実行	教T	20	正解

機能がすべて動いているか

ブループリントの設計書がある場合



関数のテスト方法 (print デバッグ)



設計書通り動いているか

テストはプログラムを組むまでの逆の方向で実施していく

ブループリントを学習する
算数ゲームを作りたい

算数ゲームの中身と
フローを書き起こす

実装しないといけない機能を
詳細に書き起こす

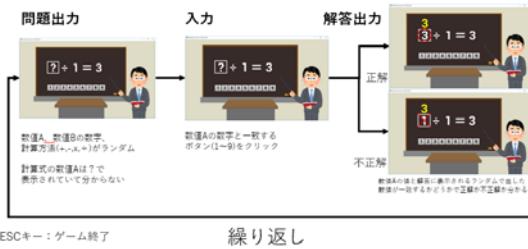
機能ごとの作り方を考える

プログラミング

算数ゲーム

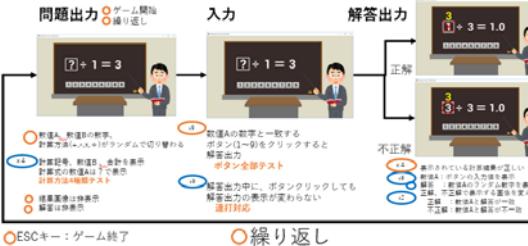
ブループリント

作成するゲーム内容



プレイしてみて
作ろうとしていた
算数ゲームが出来ているか

抽出したテスト項目



○表示に関連するテスト
○機能に関連するテスト

問題番号	実現度	実現状況	テスト名
2-1	実現済	実現済	2-1実現済み
2-2	実現済	実現済	2-2実現済み
2-3	実現済	実現済	2-3実現済み
2-4	実現済	実現済	2-4実現済み
2-5	実現済	実現済	2-5実現済み
2-6	実現済	実現済	2-6実現済み
2-7	実現済	実現済	2-7実現済み
2-8	実現済	実現済	2-8実現済み
2-9	実現済	実現済	2-9実現済み
2-10	実現済	実現済	2-10実現済み
2-11	実現済	実現済	2-11実現済み
2-12	実現済	実現済	2-12実現済み
2-13	実現済	実現済	2-13実現済み
2-14	実現済	実現済	2-14実現済み
2-15	実現済	実現済	2-15実現済み
2-16	実現済	実現済	2-16実現済み
2-17	実現済	実現済	2-17実現済み
2-18	実現済	実現済	2-18実現済み
2-19	実現済	実現済	2-19実現済み
2-20	実現済	実現済	2-20実現済み
2-21	実現済	実現済	2-21実現済み
2-22	実現済	実現済	2-22実現済み
2-23	実現済	実現済	2-23実現済み
2-24	実現済	実現済	2-24実現済み
2-25	実現済	実現済	2-25実現済み
2-26	実現済	実現済	2-26実現済み
2-27	実現済	実現済	2-27実現済み
2-28	実現済	実現済	2-28実現済み
2-29	実現済	実現済	2-29実現済み
2-30	実現済	実現済	2-30実現済み
2-31	実現済	実現済	2-31実現済み
2-32	実現済	実現済	2-32実現済み
2-33	実現済	実現済	2-33実現済み
2-34	実現済	実現済	2-34実現済み
2-35	実現済	実現済	2-35実現済み
2-36	実現済	実現済	2-36実現済み
2-37	実現済	実現済	2-37実現済み
2-38	実現済	実現済	2-38実現済み
2-39	実現済	実現済	2-39実現済み
2-40	実現済	実現済	2-40実現済み
2-41	実現済	実現済	2-41実現済み
2-42	実現済	実現済	2-42実現済み



機能がすべて動いているか

ブループリントの設計書がある場合



設計書通り動いているか

ブループリントを学習する
算数ゲームを作りたい

算数ゲームの中身と
フローを書き起こす

実装しないといけない機能を
詳細に書き起こす

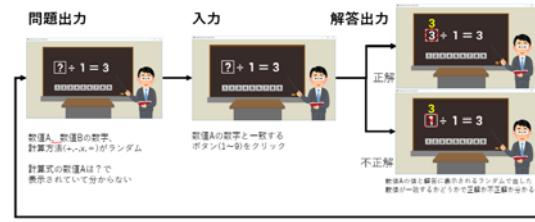
機能ごとの作り方を考える

プログラミング

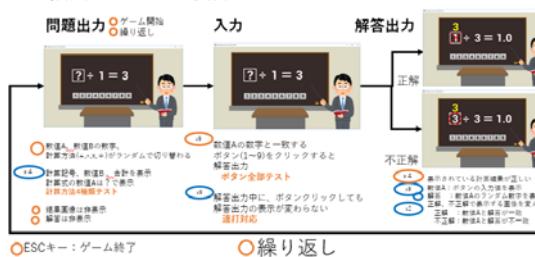
作ろうとしたゲームにもっとこうしたいと 思つたらまた、ブレークダウンしていく 算数ゲーム

ブループリント

作成するゲーム内容



抽出したテスト項目



ブループリントの設計書がある場合



配列やループを使うようにして、複数回計算して得点を出力したい



プレイしてみて
作ろうとしていた
算数ゲームが出来ているか

テスト項目票(表示: 解答出力)

問題番号	出題内容	正解	テスト用例
2-1	算算式: 1+1=2	1+1=2	1+1=2
2-2	算算式: 2+2=4	2+2=4	2+2=4
2-3	算算式: 3+3=6	3+3=6	3+3=6
2-4	算算式: 4+4=8	4+4=8	4+4=8
2-5	算算式: 5+5=10	5+5=10	5+5=10
2-6	算算式: 6+6=12	6+6=12	6+6=12
2-7	算算式: 7+7=14	7+7=14	7+7=14
2-8	算算式: 8+8=16	8+8=16	8+8=16
2-9	算算式: 9+9=18	9+9=18	9+9=18
2-10	算算式: 10+10=20	10+10=20	10+10=20
2-11	算算式: 11+11=22	11+11=22	11+11=22
2-12	算算式: 12+12=24	12+12=24	12+12=24
2-13	算算式: 13+13=26	13+13=26	13+13=26
2-14	算算式: 14+14=28	14+14=28	14+14=28
2-15	算算式: 15+15=30	15+15=30	15+15=30
2-16	算算式: 16+16=32	16+16=32	16+16=32
2-17	算算式: 17+17=34	17+17=34	17+17=34
2-18	算算式: 18+18=36	18+18=36	18+18=36
2-19	算算式: 19+19=38	19+19=38	19+19=38
2-20	算算式: 20+20=40	20+20=40	20+20=40

関数のテスト方法 (print デバッグ)



機能がすべて動いているか

設計書通り動いているか

9. パッケージ化

9. パッケージ化

9.1 配布用パッケージにするためのプロジェクト設定

9.2 Windows(64ビット)でパッケージ化

9.3 パッケージサイズをより小さくする

9. パッケージ化

9.1 配布用パッケージにするためのプロジェクト設定

9.2 Windows(64ビット)でパッケージ化

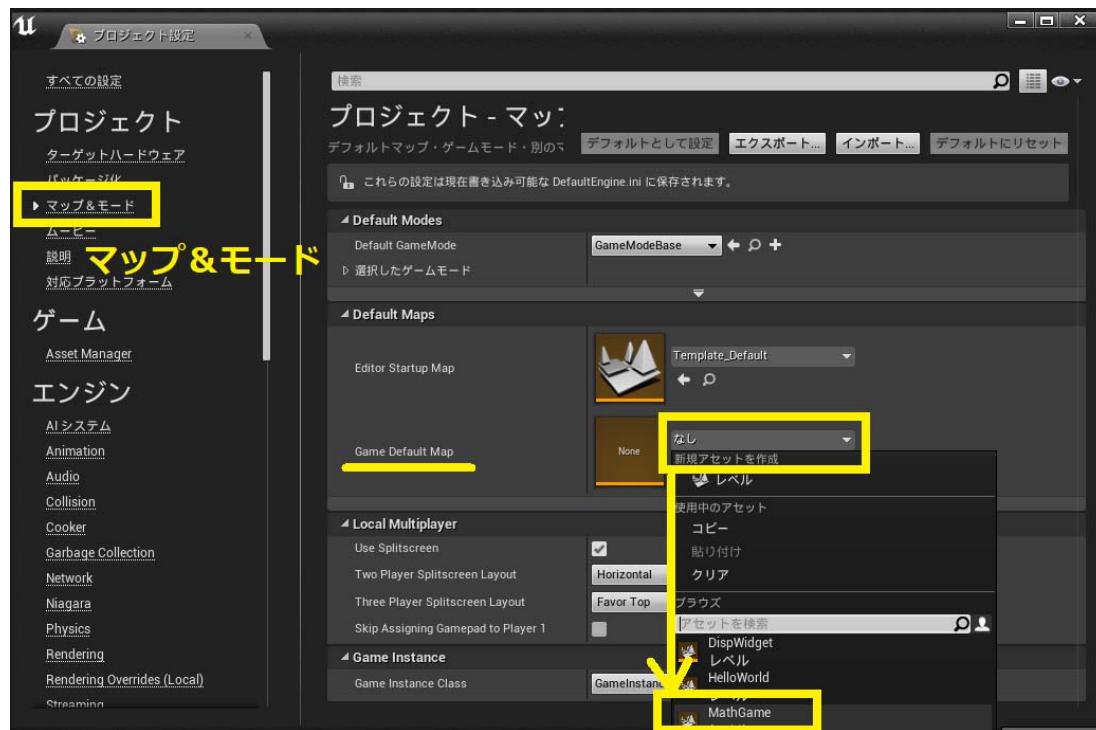
9.3 パッケージサイズをより小さくする

プロジェクト設定を開く



設定 > プロジェクト設定

ゲームを開始した時に1番最初に開くレベルを設定する

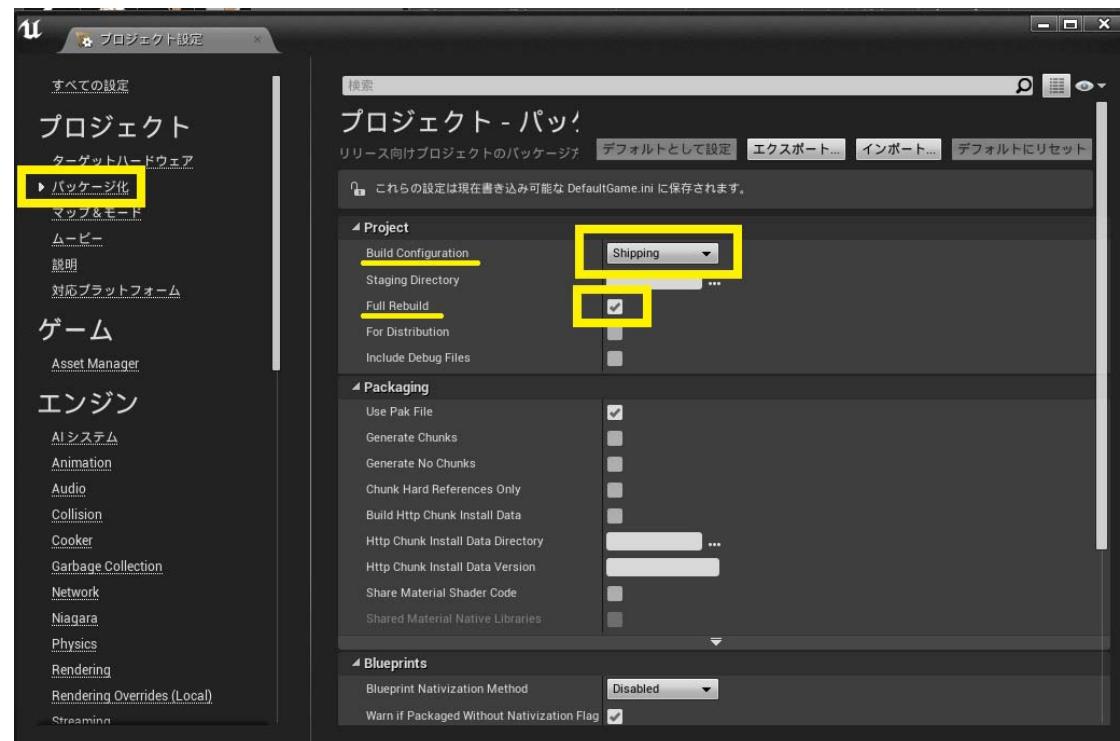


プロジェクト | マップ & モード

項目	設定
GameDefaultMap	MathGame

Game Default Mapは
ゲームを開始した時に1番最初に開くレベルを設定する

パッケージを出荷用のビルド設定にする



プロジェクト | パッケージ化

項目	設定
Build Configuration	Shipping
Full Rebuild	チェック

プロジェクト | パッケージ化を選択
Build Configuration, Full Rebuildを変更する

9. パッケージ化

9.1 配布用パッケージにするためのプロジェクト設定

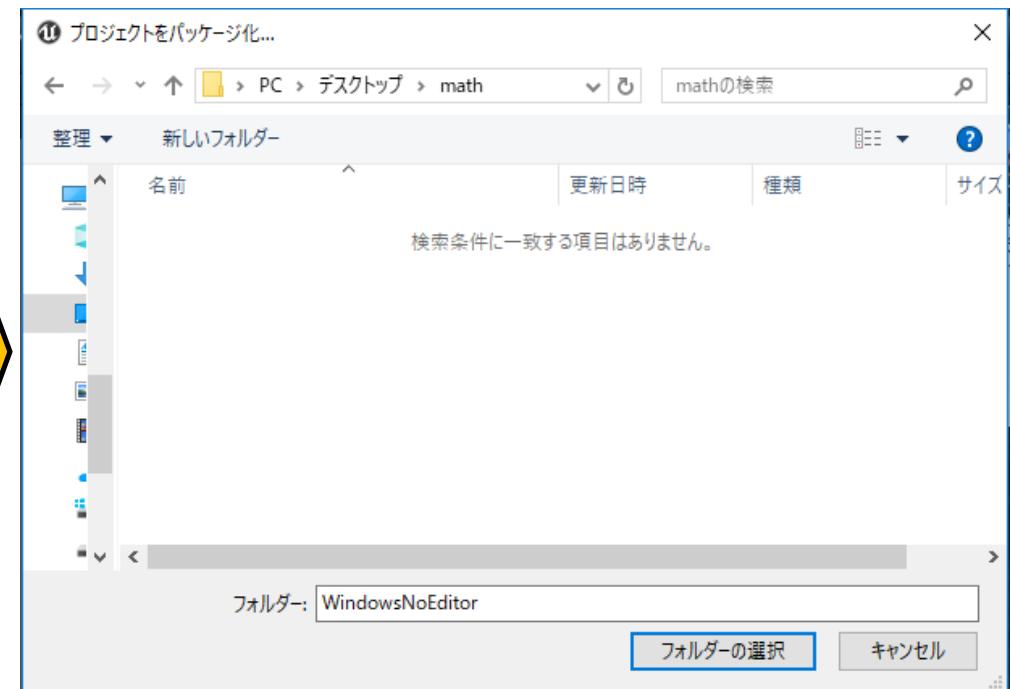
9.2 Windows(64ビット)でパッケージ化

9.3 パッケージサイズをより小さくする

プロジェクトをパッケージ化 (Windows(64ビット))

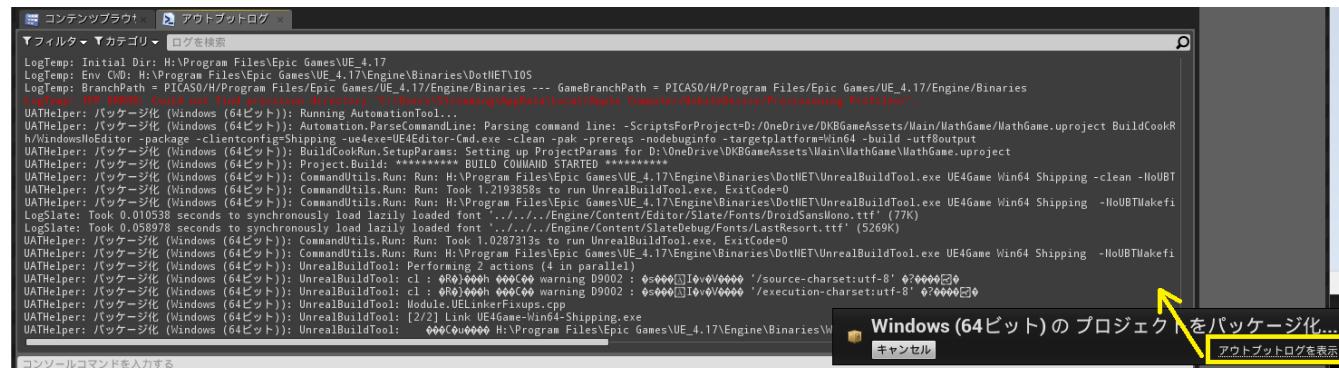


ファイル > プロジェクトをパッケージ化
> Windows > Windows(64ビット)

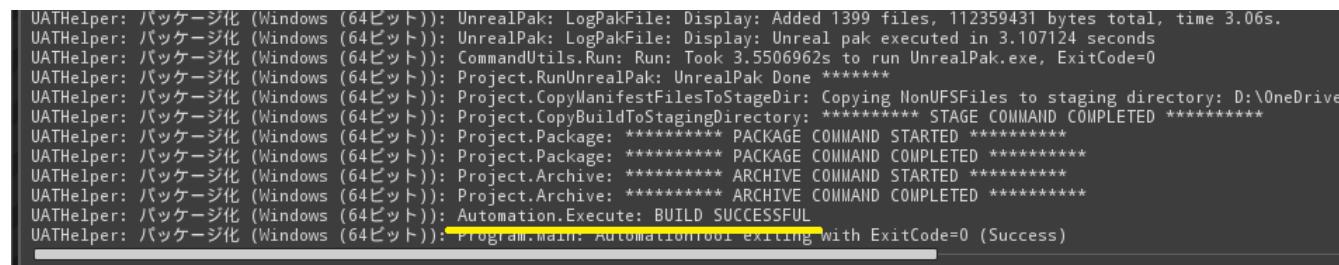


パッケージを書き出すフォルダを選択
> フォルダの選択

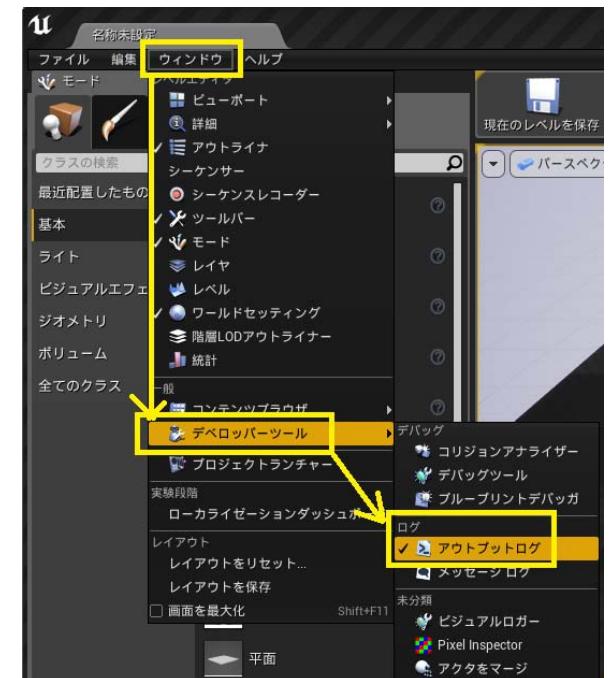
アウトプットログウィンドウでビルド状況を確認



右下にダイアログが表示される
アウトプットログを表示をクリックすると
アウトプットログウィンドウが表示される

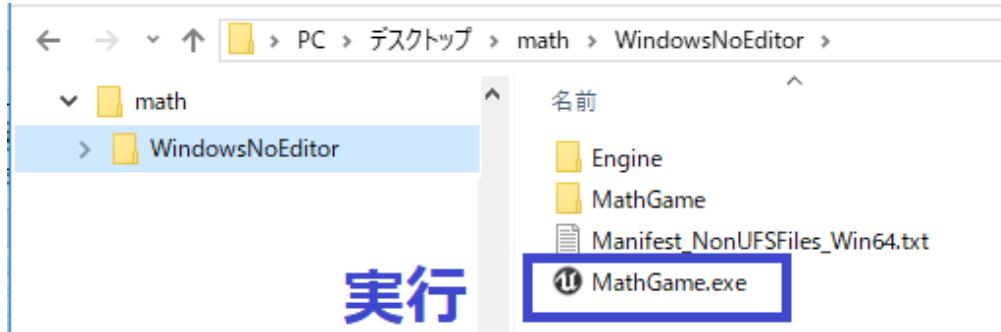


BUILD SUCCESSFULが表示されればパッケージ化成功

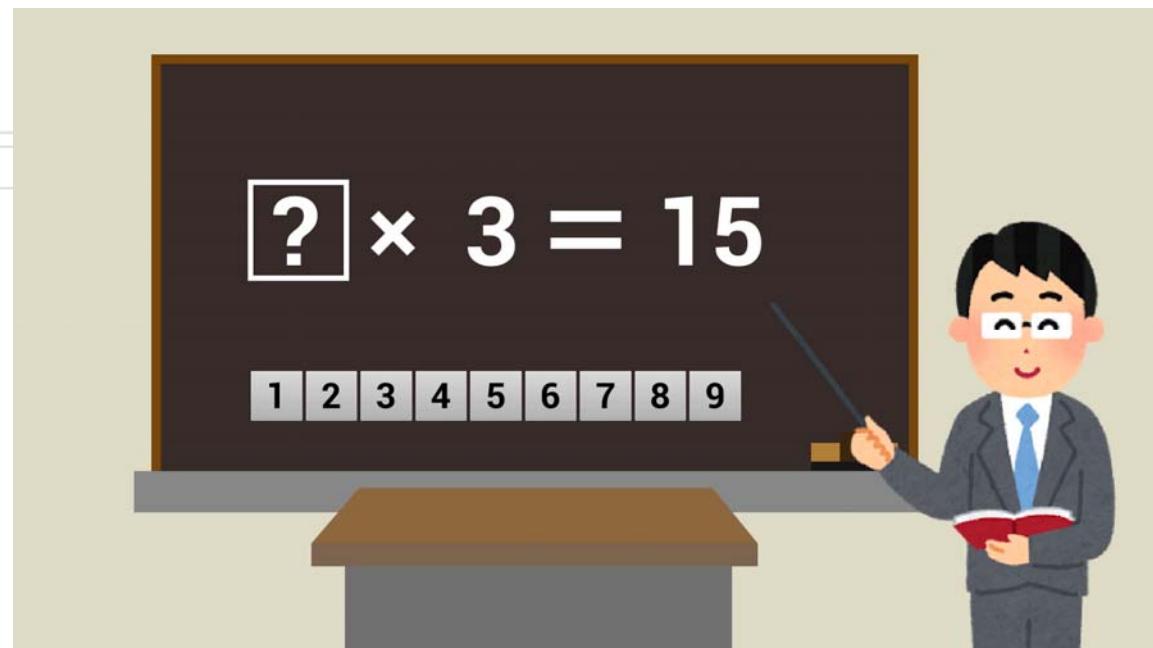


アウトプットログの表示手順
ウィンドウ > デベロッパーツール
> アウトプットログ

選択したフォルダにWindowsNoEditorフォルダが作成され、実行ファイルが出力される



選択したフォルダにWindowsNoEditorフォルダが作成される
MathGame.exeを実行する



ゲームが開始される
WindowsNoEditorフォルダ配下を全部コピーすれば、
他のPCでも実行することが出来る

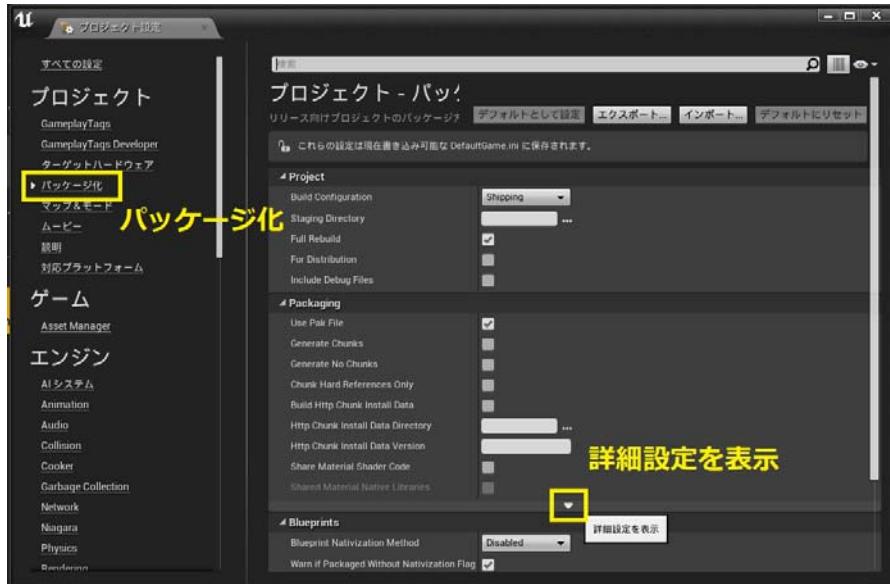
9. パッケージ化

9.1 配布用パッケージにするためのプロジェクト設定

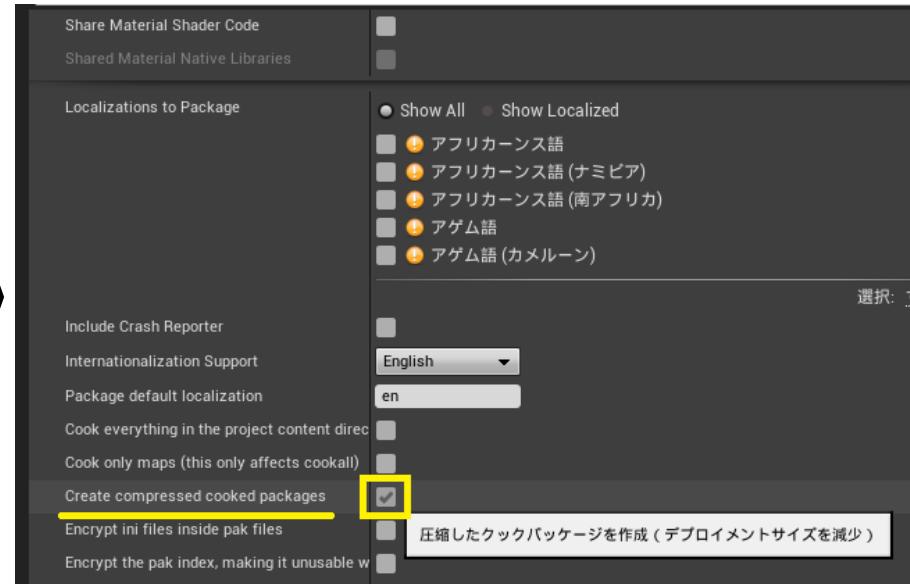
9.2 Windows(64ビット)でパッケージ化

9.3 パッケージサイズをより小さくする

クックしたパッケージを圧縮する（1番小さくなる） Create compressed cooked packagesにチェック



パッケージ化 > Packagingカテゴリの▼をクリック



Create compressed cooked packageにチェック

合計210 MB > 138 MBまで減った

パッケージ化したゲームのサイズを小さくする方法（他にも色々ある方法があるが、自己責任で）

<https://docs.unrealengine.com/latest/JPN/Engine/Performance/ReducingPackageSize/index.html>