ポートフォリオ

名古屋工学院専門学校　藤嶋　翔太

Team Sunset

自己紹介

名前　ふじしま　しょうた　藤嶋　翔太

所属　名古屋工学院専門学校
ゲーム総合学科
ゲームプログラミングコース

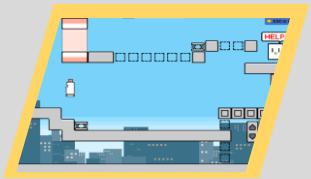
誕生日　1999年　8月31日

使用環境　エンジン:Unity　言語:C#

趣味　ゲームコラボアクセサリ集め
料理(主にバイト先にてメニュー
以外で作れそうな料理の開発)

自己紹介/制作したゲーム

１年次



２年次

３年次

４年次

モーション検証用
２本

制作したゲーム
１０本

ゲーム紹介

ゲーム紹介/TANK FRONT LINE 3

タイトル

TANK FRONT LINE 3

ジャンル

３Dアクションシューティング

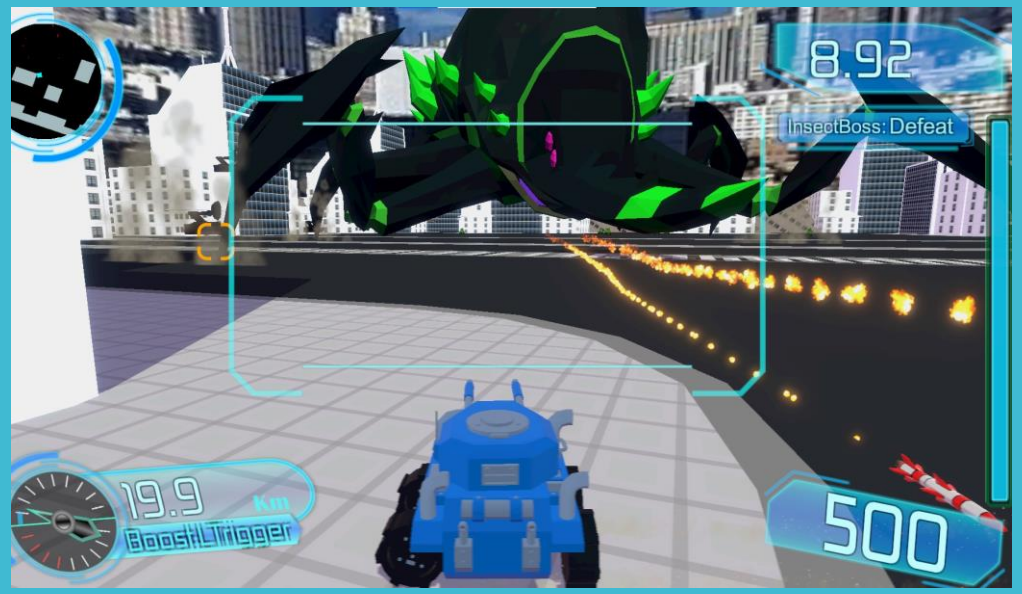
制作人数

３人(内訳：プログラマ３人)

制作期間

2023年2月　～　2023月5月末
(約73日 372時間)

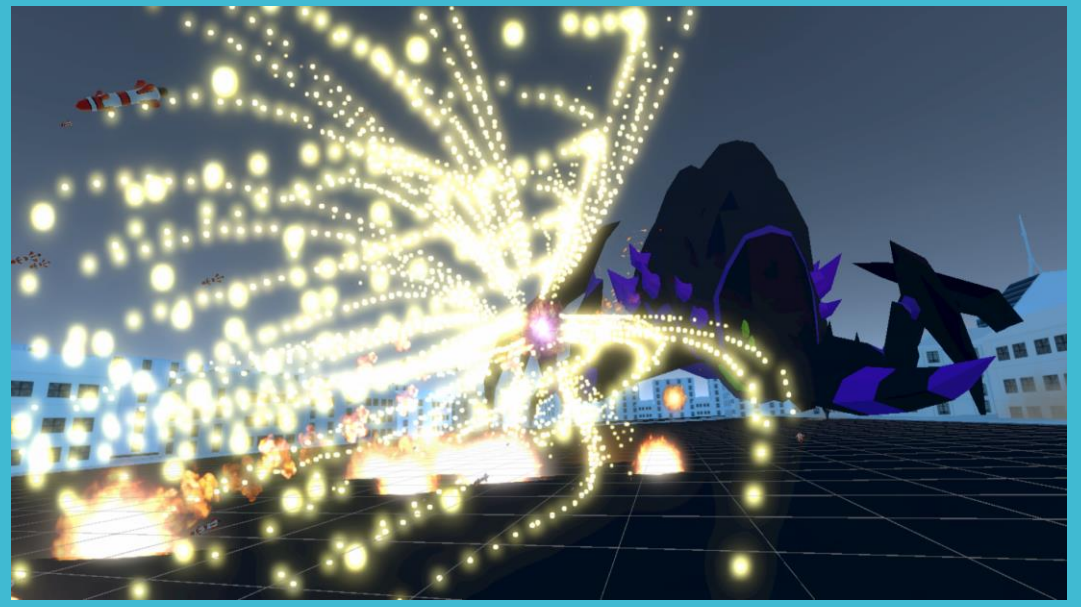
ゲーム紹介/TANK FRONT LINE 3



日本ゲーム大賞アマチュア部門2023に向け制作したゲームです。
左輪を左スティックで、右輪を右スティックで操作し、様々な種類のミッションをクリアしていくゲームとなっております。
私のこだわりポイントは操作方法とミッション選択時の演出です。

本作のミサイルはUnity既存のメソッドに頼らずに内積外積を用いて、角度計算を行っております。
理由はC#以外の言語に移った場合でも数式さえ知っていれば角度計算等は可能だと判断したためです。

※　日本ゲーム大賞アマチュア部門2023 2次審査突破
　　ゲームクリエイター甲子園2023 アプシィ賞　授賞



ゲーム紹介/SHI KA KU

タイトル

SHI KA KU

ジャンル

ステルスパズルアクション

制作人数

２人(内訳：プログラマ２人)

制作期間

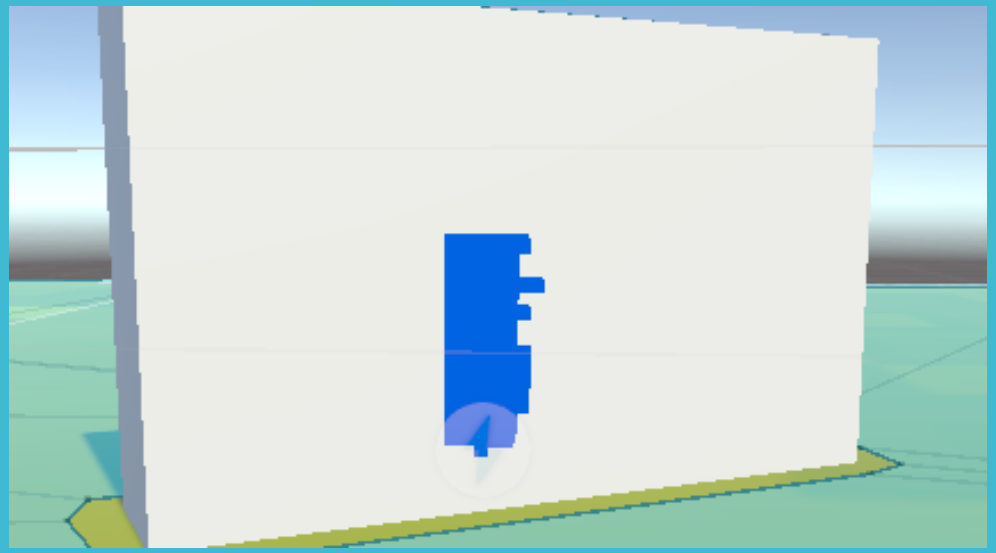
2024年9月　～　2024年11月末
(約63日　207時間)

ゲーム紹介/SHI KA KU

「しかく」をテーマに制作したゲームです。
キャッチコピーは
「四角いやつらの死角から刺客を送り込め！」
敵の視界に入らないよう動物を駆使して敵の死角に回り込んで倒していくゲームです。
こだわりポイントは「しかく」をゲームに落とし込んだ部分と雰囲気を柔らかくすることでプレイまでのハードルを下げた部分です。

今作ではスタンダードサーフェスシェーダの調整を行いシルエット表示できるようにしました。
理由は壁裏などに移動した際にプレイヤーや敵の位置が分からなくなると困ること、そして一度シェーダに触れてみようと考えたためです。

※　ゲームクリエイター甲子園2024 応募中
サイゲームスクリエイティブコンテスト2024 応募中



ゲーム紹介/使用環境について

開発環境、仕様ツール等

Unity ver.2021.3.12f1

UniRx

GItHub,SourceTree

開発環境、仕様ツール等

Unity ver.2023.2.20f1

R3

UniTask

DoTween

GItHub,SourceTree

工夫点

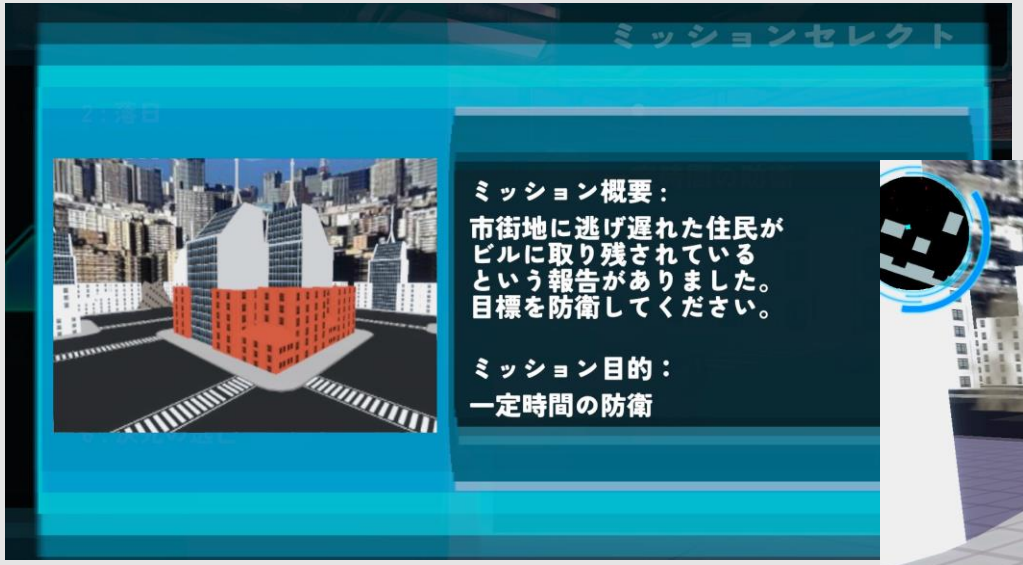
UniRxの導入

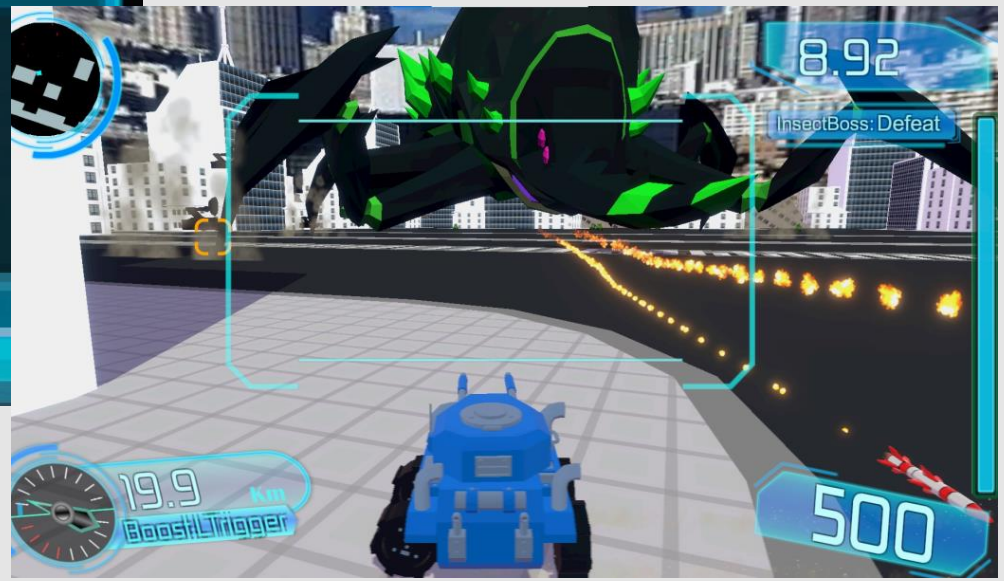
UniRxを導入し、MVPパターンの形にすることでプレイヤーのHPなどのパラメータ管理とUIのHPなどの連動する部分を切り離し、お互いが存在しないとエラーが出てしまう状況を回避しました。

ステージのメリハリ

同じタイプのミッションが続くと、「またか…」となる可能性があるため、

ステージのメリハリ





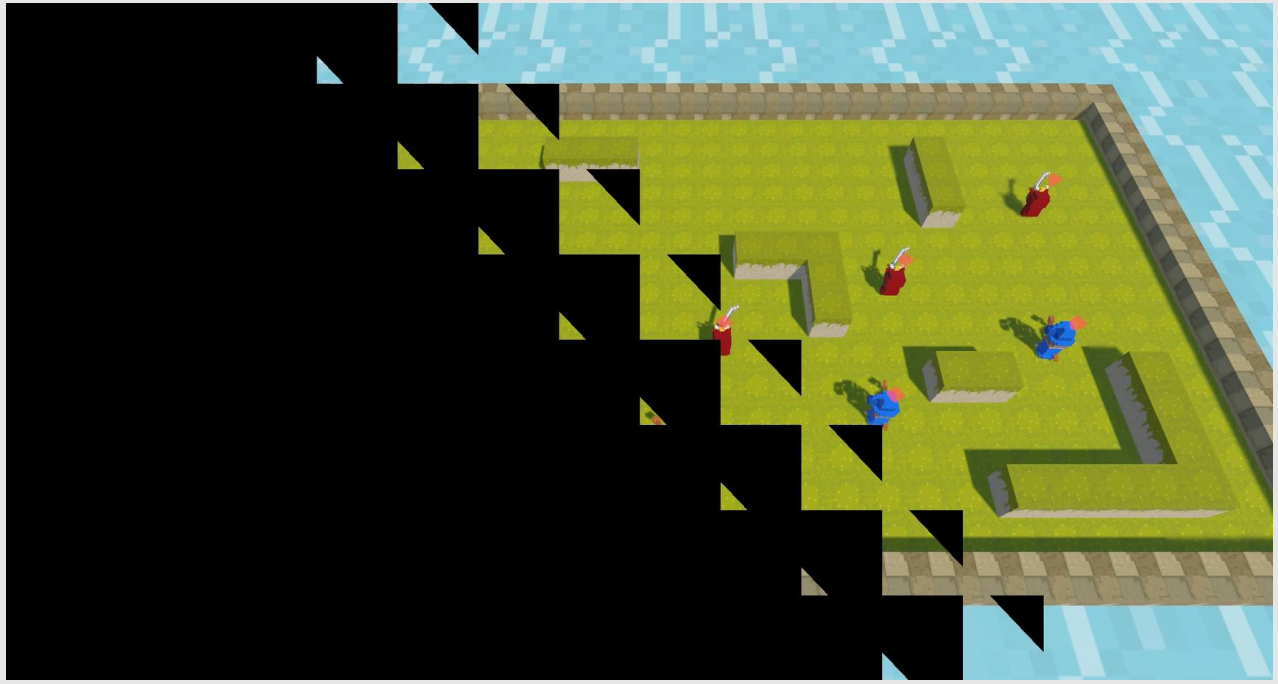
特定施設の防衛やボス戦を合間に絡めることで
ミッション毎のメリハリ感を演出しました。

テーマに沿ったフェード

```
float _AspectX;
float _AspectY;
int _SplitSize;

float4 _MainColor;

vector _DivideScreen;
float _AnimationTime;

float slide(float2 uvFloat,int2 uvInt,float time)
{
    return step(time + 2.0 , uvFloat.x - (uvInt.x - time) + uvFloat.y - (uvInt.y - time));
}

float4 frag (v2f i) : SV_Target
{
    float2 uvFloat = float2(frac(i.uv.x * _DivideScreen.x),frac(i.uv.y * _DivideScreen.y));
    int2 uvInt = int2(floor(i.uv.x * _DivideScreen.x),floor(i.uv.y * _DivideScreen.y));

    float alpha = step(_AnimationTime,uvFloat.x + uvFloat.y);

    return float4(0,0,0, slide(uvFloat,uvInt,_AnimationTime));
}
ENDCG
```

よくある画面全体を暗くしていくフェード処理では他のゲームとの差別化が出来ないと考えたため、上記のシェーダを作成しました。

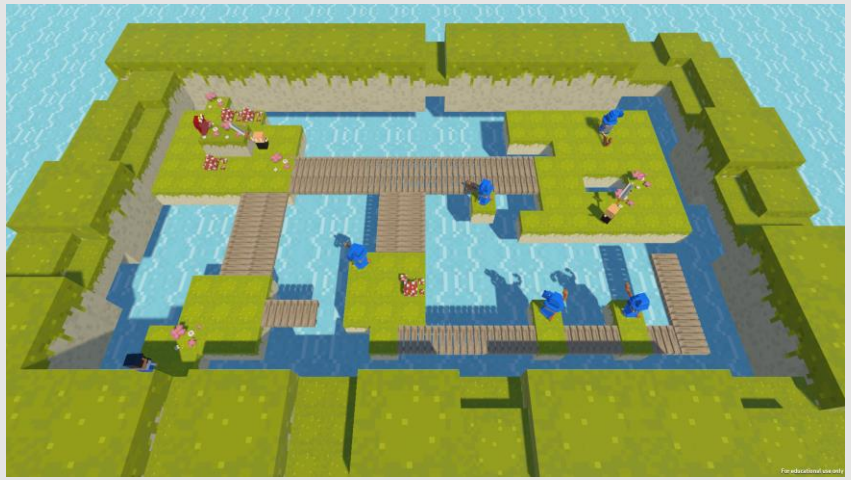
テーマに沿ったフェード



実際に動く動画はこちらから

このシェーダにより今作のテーマ「しかく」に合った
フェード処理を作ることが出来ました。

ステージ開始時の演出



```
vector _ExpandStartPos;
float _AnimationTime;

float Expand(float2 uv,float animationTime)
{
    float distanceValueX = distance(_ExpandStartPos.x,uv.x) * (16.0/9.0);
    float distanceValueY = distance(_ExpandStartPos.y,uv.y);

    float convert = step(distanceValueX,_AnimationTime) * step(distanceValueY,_AnimationTime);

    return abs(convert - 1);
}

float4 frag (v2f i) : SV_Target
{

    return float4(0,0,0,Expand(i.uv,_AnimationTime));

}
ENDCG
```

ゲーム開始時にプレイヤーの初期位置が分かりにくいという問題点が出てきたため、上記のシェーダを作成しました。

ステージ開始時の演出

実際に動く動画はこちらから

ステージ開始時にプレイヤーだけをピックアップしてから全体を映すようにするフェードを実装することで初期位置の分かりにくさを改善しました。

ステージセレクトの視覚的な楽しさ

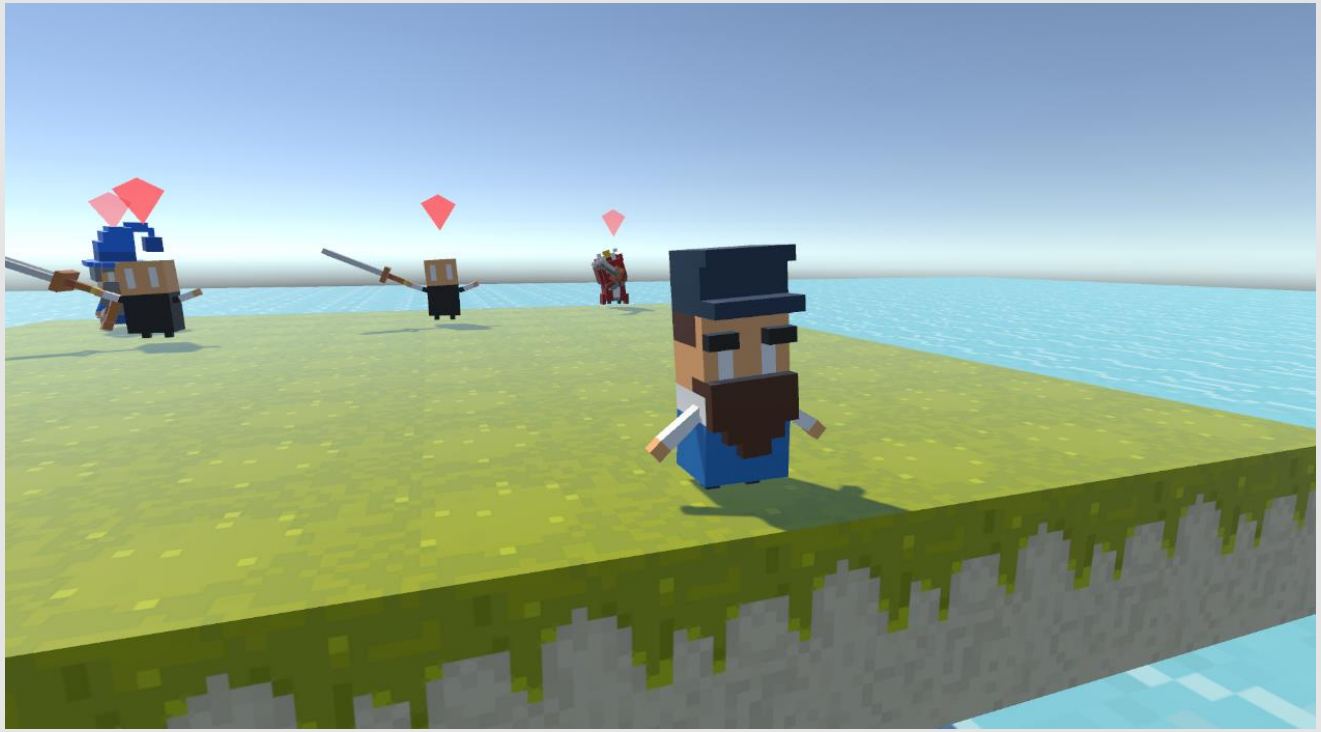
画面全体が静止した状態なため、見た目における
楽しさがかなり薄いと感じたため、

ステージセレクトの視覚的な楽しさ

実際に動く動画はこちらから

各ワールドのモチーフオブジェクトを回転させるなどの視覚的に動きをつけることで、見た目における楽しさを演出しました。

モーションが足りない！！



作成したモーションはこちらから

各キャラクターに必要なモーションがありませんでした。そのため、Blenderを用いて必要なモーションを自作しました。

技術検証

技術検証/モンチェイス

タイトル

モンチェイス

ジャンル

アクション(Unreal技術検証)

制作人数

２人(内訳：プログラマ２人)

制作期間

2024年2月　～　2024年5月
(約33日　60時間)

ステート実装

StateBase.cpp

```
void UStateBase::OnEnter()
{
}

void UStateBase::OnUpdate()
{
}

void UStateBase::OnExit()
{
}

void UStateBase::Bind()
{
}

void UStateBase::UnBind()
{
}
```

ベースのステートクラスに基本のEnter、Update、Exit以外にステートごとの行動をボタン追加、削除するためのメソッドを実装

ステート実装

StateBase.h

```
UInputComponent* inputComponent = nullptr;
```

MainPlayerクラスの継承元であるACharacterの持つUInputComponentを取得し、各ステートが参照を持つことで各ステートでの行動の追加、削除を可能にしている

Animation BluePrintとの連携

PlayerAnimInstance.h

```
UCLASS()
class PROJECTAPE_API UPlayerAnimInstance : public UAnimInstance
{
    GENERATED_BODY()

public:
    virtual void NativeBeginPlay() override;

    UPROPERTY(BlueprintReadOnly)
    bool isRun = false;

    UPROPERTY(BlueprintReadOnly)
    bool isJump = false;
    UPROPERTY(BlueprintReadOnly)
    bool isDoubleJump = false;
    UPROPERTY(BlueprintReadOnly)
    bool isFall = false;
};
```

継承

必要な変数を追加した独自のインスタンスを作り、
それを継承したAnimationBPを作る

Animation BluePrintとの連携

MainPlayer.cpp

```
81      // アニメーション情報をステートジェネレーターに渡す
82      animBPInstance = Cast<UPlayerAnimInstance>(skeletalMesh->GetAnimInstance());
83      stateGenerator->animBPInstance = this->animBPInstance;
```

スケルタルメッシュコンポーネントに設定されているAnimationBPを取得し、ステートジェネレータを通じて各ステートに渡す

Animation BluePrintとの連携

MoveState.cpp

```
void UMoveState::OnEnter()
{
    UKismetSystemLibrary::PrintString(this, "MoveEnter");
    Bind();
    animBPInstance->isRun = true;
}
```

C++上でboolの値を変更

C++上でbool値の変更出来るようにすることで
・ステートパターンで行えるEnter時、Exit時に処理を行える
AnimationBPを使うことで
・アニメーションのブレンドを使うことが出来る
・視覚的に各アニメーションの繋がりが見やすい
上記のメリットを獲得することが出来る

ポートフォリオサイトなど

YouTube個人チャンネル
https://www.youtube.com/channel/UCaAEZyrNl6Up2VO6BgHcdbg

Vivivit デザイナーポートフォリオサイト
https://www.vivivit.com/portfolio/public/6VmHZ5nmKA