アルケミフォートレス
作品情報　アピールポイント

トライデントコンピュータ専門学校　３年
制作者　長瀬和摩

目次

アルケミフォートレス

フォルダ構成

プログラム作品

実行形式

ソースコード

アピールポイント(本ファイル)

ゲーム解説動画

アルケミフォートレス

フォルダ構成

ソースコード

NecromaLib　自作のフレームワーク

packages　DirectXTK

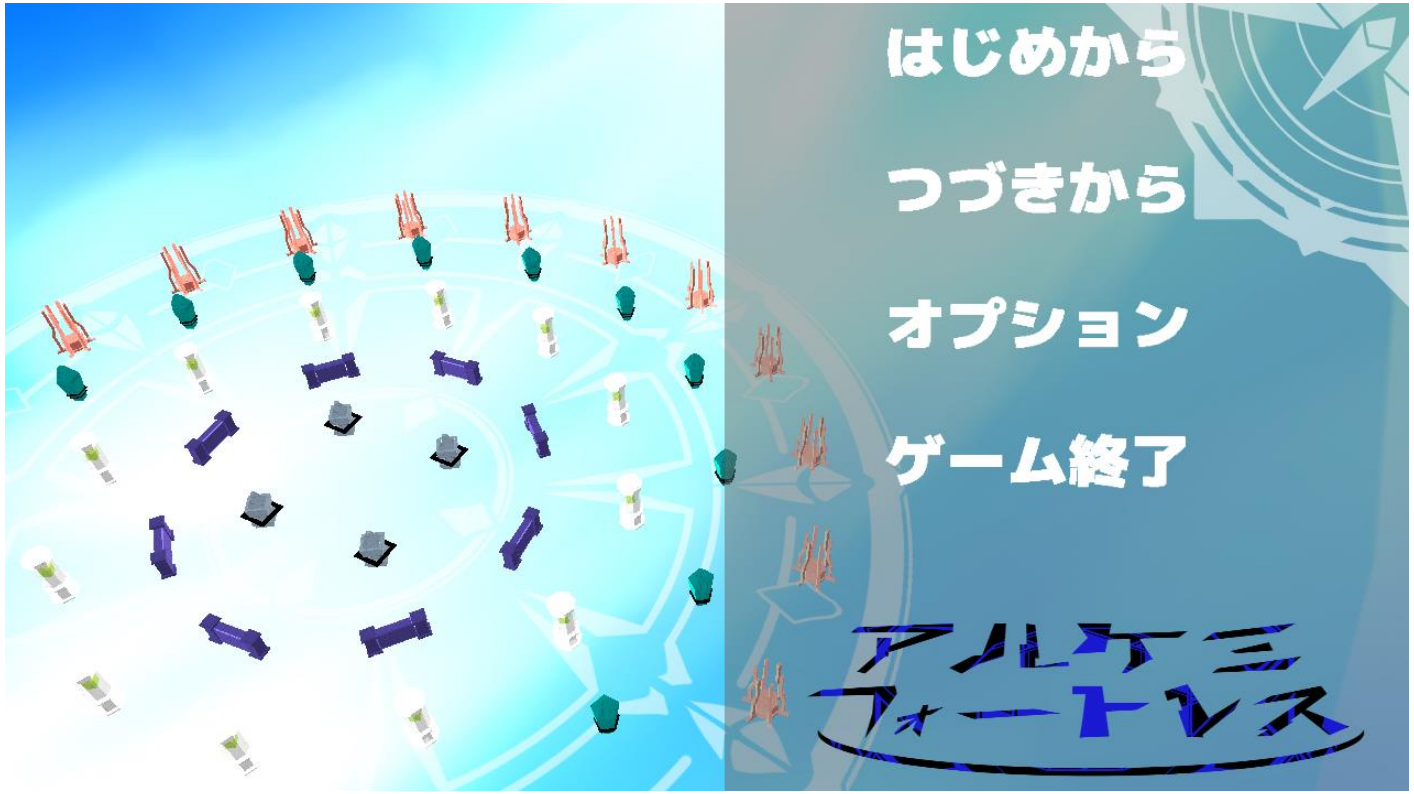
Resources　音 画像 3Dモデル素材
hlsl,jsonファイルなど

Scenes　シーン毎に分けたプログラムフォルダ

AlchemiFortress

作品情報

ジャンル　　：3Dタワーディフェンス
プレイ人数　：1人
制作期間　　：2023/6~2024/1
制作人数　　：1人
担当範囲　　：プログラム,モデル,BGM全般
開発環境　　：DirectX11 DirectXTK
動作環境　　：Windows11



ゲーム概要

限られたリソース内でユニットを配置し敵から拠点を守るタワーディフェンス

数秒後のユニット,敵の位置を予測して配置,回転停止をすることがカギとなる

アルケミフォートレス

操作方法

基本操作はマウスのみ

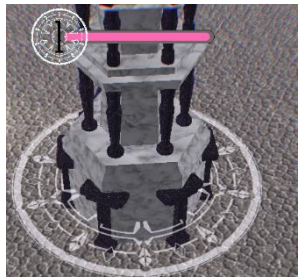
ホイール

スクロール

ズームイン/アウト

押し込み

カメラ移動

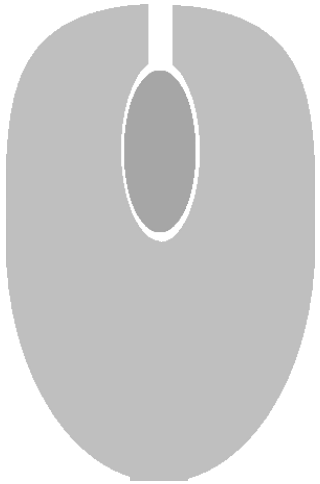


画面右下に
操作補助完備

左クリック

決定

UIやオブジェクトの決定アクション



右クリック

回転状態の切り替え

回転中

回転停止

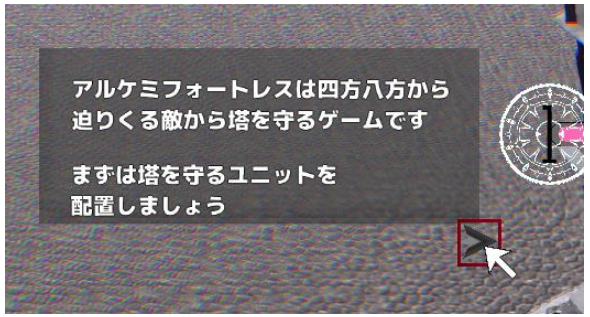
アルケミフォートレス

制作意図

複雑なシステムでも伝わるゲームを作りたい

チュートリアル UI 演出 画面効果を効果的に織り交ぜ、
複雑なゲームシステムであっても
ユーザーに優しい、伝わるゲームを作りたいと考え、制作を行いました。
またそれらを実装するにあたり、開発がしやすくなる工夫も行いました。

チュートリアル



枠線や視線誘導でユーザーを補助

UI

選択できないものは より分かりやすく
明度を下げ、選択された際には震える
アクションを実装

演出

ユニット設置時のディゾルブ表現+
パーティクル

プレイヤーが起こした
アクションに対して必ず
フィードバックを行うように徹底

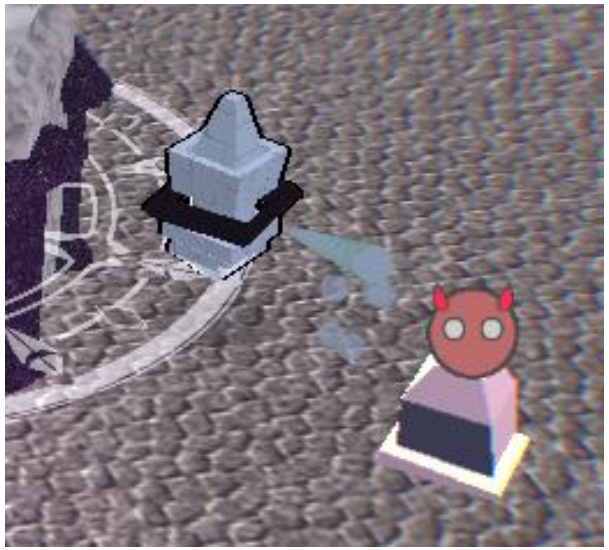
アルケミフォートレス

ゲーム概要

ユニットを錬金/設置して敵から塔を守ろう

左クリックでユニットを錬金

設置可能な場所に自動で設置

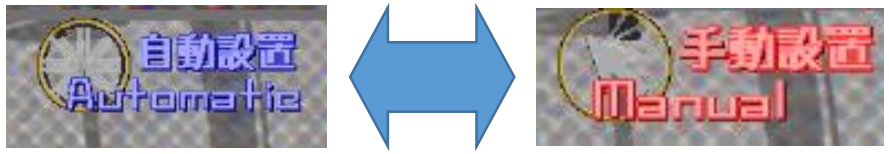


魔力リソースを消費して
範囲内の敵を自動で攻撃

錬金には各リソースを消費します

ゲーム概要

選べる設置モード



クリックで切り替えが可能

自動設置
Automatic

自動で設置するモード

手動設置
Manual

設置可能箇所に手動で設置するモード

ゲーム概要

本作における駆け引き

プレイヤーは任意で回転移動を止めることができる（右クリック）

回転を止めると結晶リソースを回収しやすくなったり、敵に攻撃を当てやすくなる

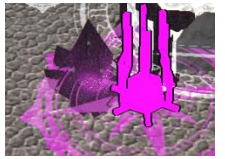
しかし、魔力リソースが回復しなくなるため、戦況を見極める必要がある

リソース回収条件

魔力：  回転移動をしている

結晶：  範囲内に結晶がある

パーティクルが出現する



紫色に変化する

外部ファイル読み込み

採用理由　・手入力によるパラメータ調整が容易　・コードを短く記述可能
・再コンパイルの必要がなくなる　・開発速度の向上が見込める為

例 UI設定

```
{
  "UILAYOUT": [
    "MASTER": {
      "POS_X": 325.0,
      "POS_Y": -65.0,
      "RAGE_X": 0.75,
      "RAGE_Y": 0.75,

      "OPTION": [
        {
          "TAG": "LAYER",
          "VAL": 1.0
        }
      ],
      "KEYS": [
        {
          "CODE": "0x41"
        },
        {
          "CODE": "0x25"
        }
      ]
    }
  ]
}
```

位置
大きさ

その他設定値

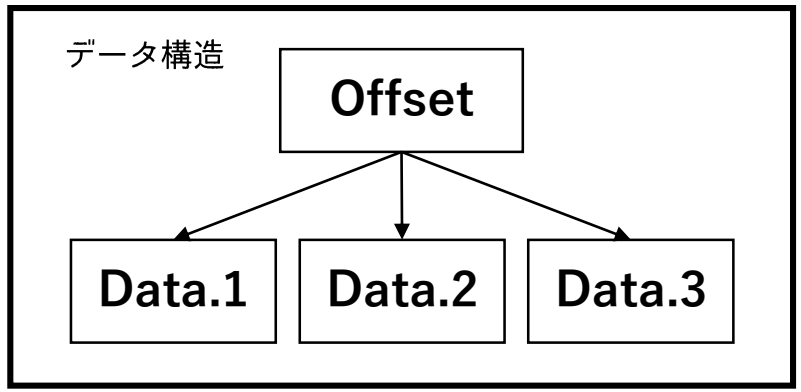
キー設定

UILayout_AlchemiMode.json

＋

```
{
  "UILAYOUT": [
    "MASTER": {
      "POS_X": 1200.0,
      "POS_Y": 640.0,
      "RAGE_X": 0.0,
      "RAGE_Y": 0.0,

      "OPTION": [
      ],
      "KEYS": [
      ]
    }
  ]
}
```

UILayout_AlchemiOffset.json




```
//  ====================[  設置モードUIの情報を取得  ]
uiData = pSJD.GetUIData("AlchemiMode");
m_modeChangeButton = std::make_unique<SelectionBox>(uiData.pos, uiData.rage);
m_modeChangeButton->Initialize();
m_modeChangeButton->SetLayer((int)uiData.option["LAYER"]);
m_modeChangeButton->SetKey(uiData.key);
```

MachineSelectManager.cpp

描画

関連フォルダ　Resources/Json

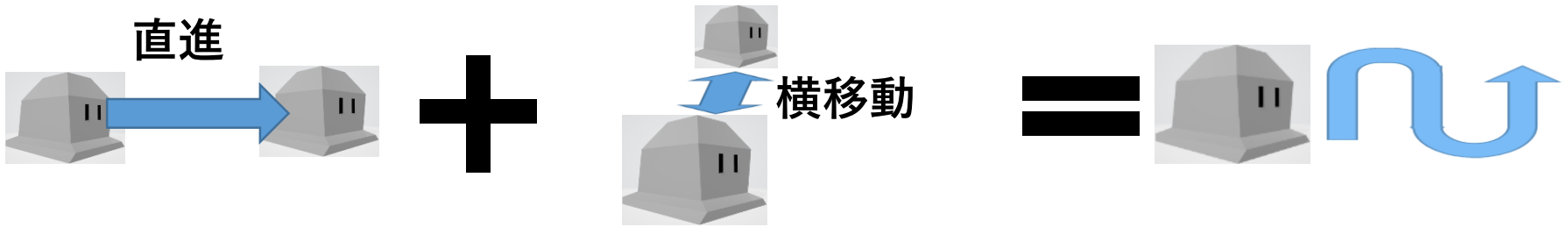
デザインパターン

コマンド.1

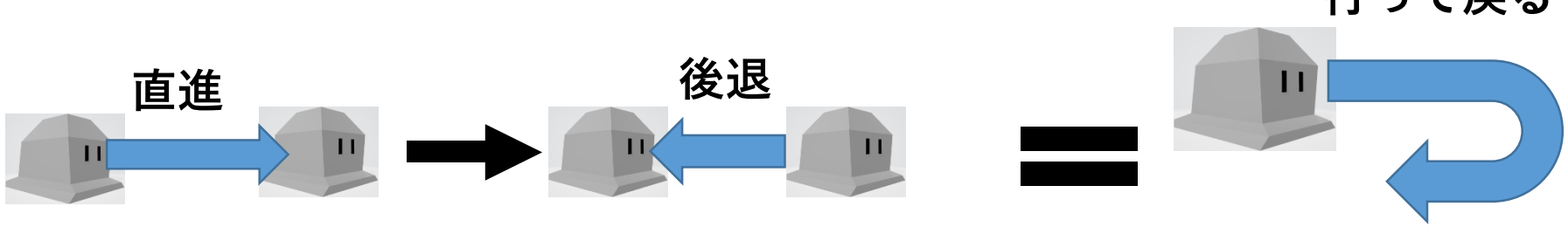
採用理由
・処理の合成,遅延実行が可能　・複数コマンドを作ることで様々な動きが作成可能
・オブジェクトとメゾットを切り離せる　・単一責任の原則,開放閉鎖の原則より

採用箇所　・エネミーの行動　・チュートリアルの処理

合成



遅延



デザインパターン

コマンド.2

動きのコマンドを外部ファイルから取得

```
{
  "Status": {
    "ELEMENT":"Aqua",
    "TYPE":"Slime",
    "MOVETYPE":"ONE",
    "HP": 10.0,
    "STR": 1.5,
    "EXP": 20.0,
    "MOVEING":[
      {
        "TYPE": "Standard",
        "TIME": 3.0,
        "DELAY": 0.0,
        "VALUE": 1.0
      },
      {
        "TYPE": "Standard",
        "TIME": 1.0,
        "DELAY": 0.0,
        "VALUE": -1.5
      }
    ]
  }
}
```

コマンド

EnemyData_Retreat.json

動きのコマンドを生成,登録

```
for (auto& moveData : data.moveData)
{
    // 受け取りたい動きの入ったコマンドクラスを取得する
    ICommand_Enemy* command = manager->CreateEnemyMoveCommand(moveData.moveName);
    // 値取得
    MoveParameter moveParam = MoveParameter();
    moveParam.delay = moveData.delay;
    moveParam.time  = moveData.time;
    moveParam.value = moveData.value;
    // コマンドクラスにパラメータを入れる
    command->SetParam(moveParam);
    // 要素分順番に入れる
    m_moveCommands.push_back(command);
}

// コマンダークラスにコマンドを登録する
for (auto& command : m_moveCommands)
{
    m_commander->AddCommand(command);
}
```

EnemyObject.cpp

登録したコマンドを実行

```
void EnemyCommander::Execute_One()
{
    // 現在のコマンド番号
    int counter = 0;

    // 稼働時間を満たしていたら加算
    for (auto& command : m_commands)
    {
        counter += command->GetCompletion();
    }

    // コマンド量より多くなった場合は0に戻す
    if (m_commands.size() <= counter)
    {
        // 全ての稼働完了済みコマンドをリセットする
        for (auto& command : m_commands)
        {
            command->SetCompletion(false);
            command->Reset();
        }

        counter = 0;
    }

    // 実行
    m_commands[counter]->Execute();
}
```

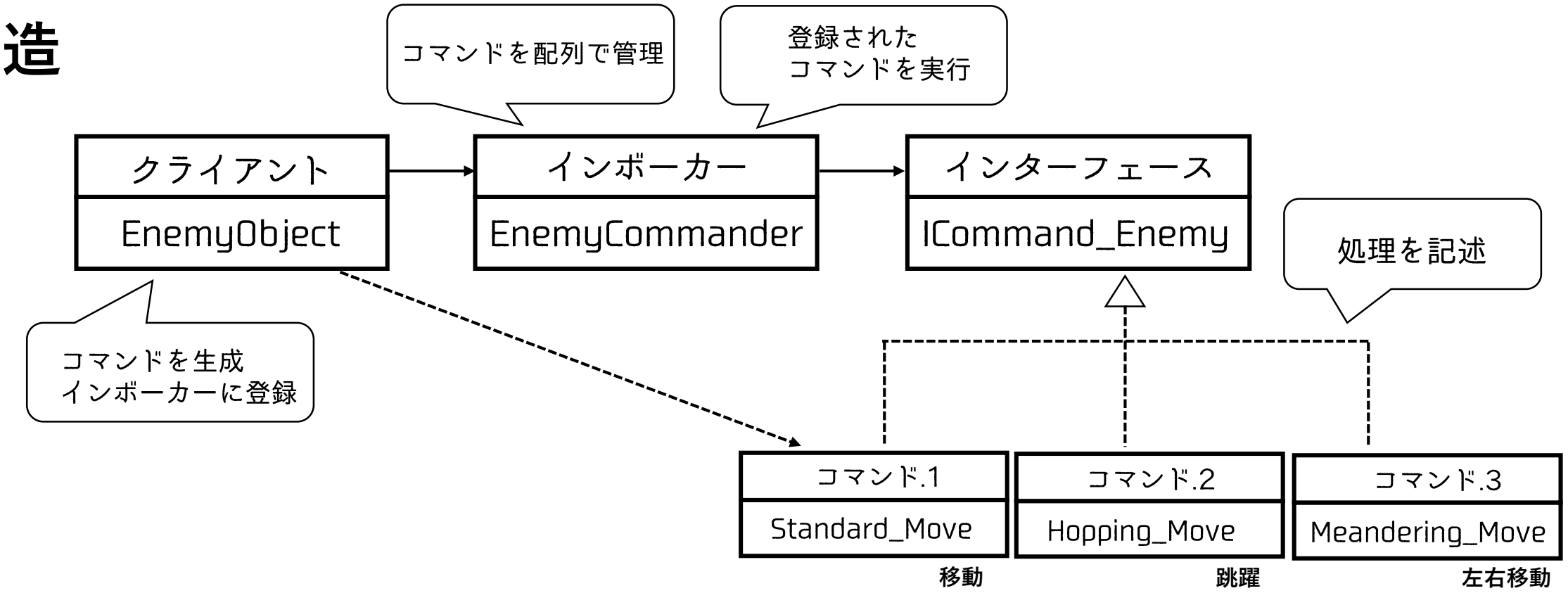
EnemyCommander.cpp

StandardMove（直線移動）
3秒前進
StandardMove
1秒後退
⋮

関連フォルダ　Scenes/PlayScene/Enemy
Scenes/PlayScene/Tutorial

デザインパターン

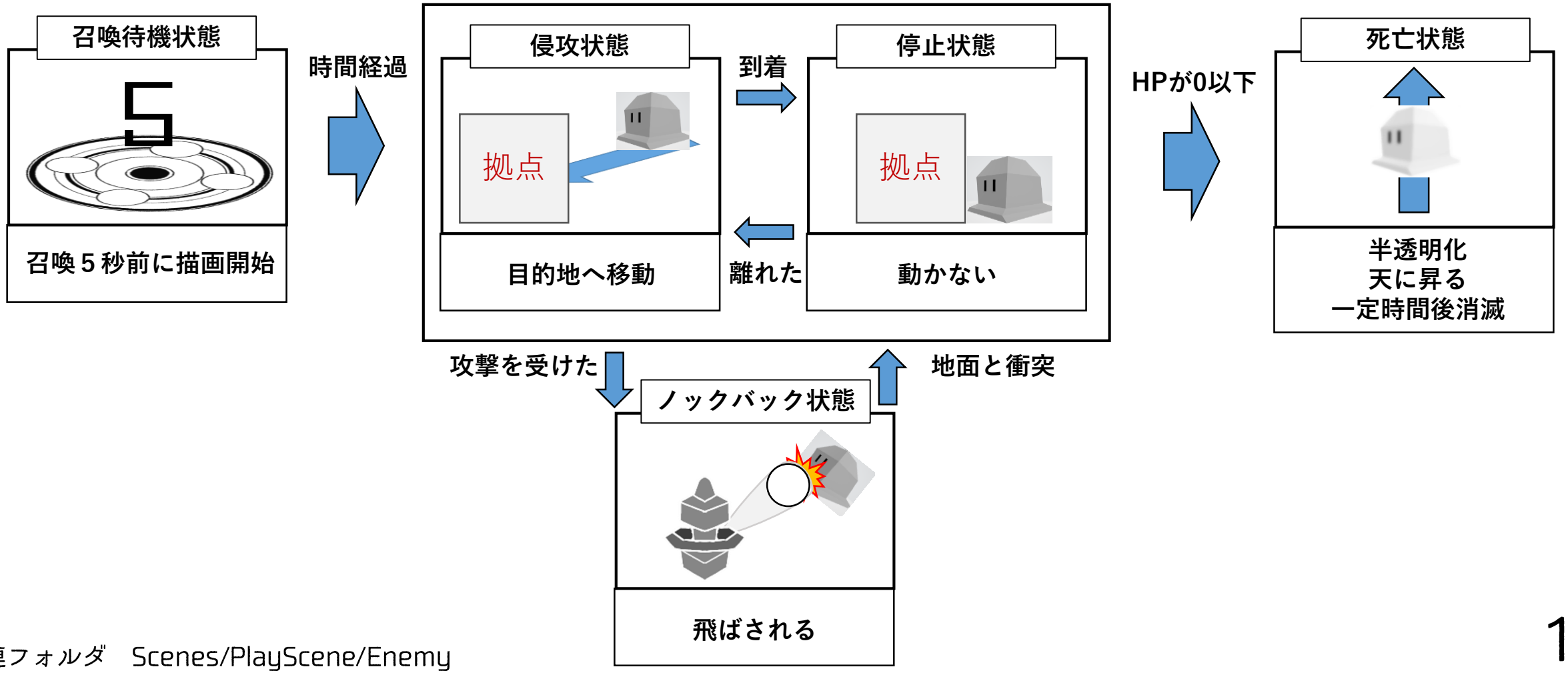
構造


コマンドを配列で管理
登録された
コマンドを実行
クライアント
EnemyObject
インボーカー
EnemyCommander
インターフェース
ICommand_Enemy
処理を記述
コマンドを生成
インボーカーに登録
コマンド.1
Standard_Move
移動
コマンド.2
Hopping_Move
跳躍
コマンド.3
Meandering_Move
左右移動

デザインパターン

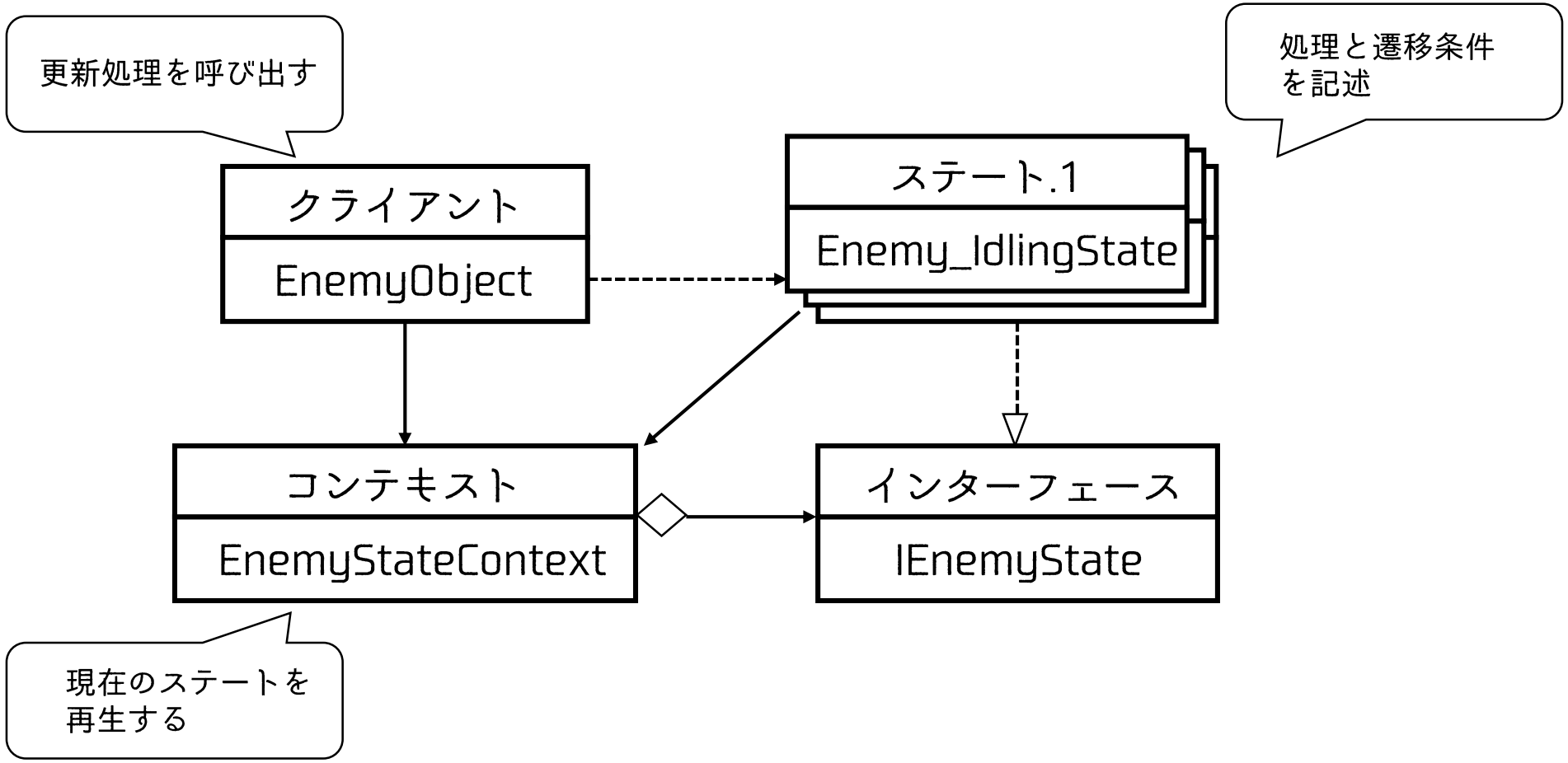
ステート.1

採用理由　・かさばった条件文を削減可能　・エネミーの状態が頻繁に状態が変化するため
・単一責任の原則,開放閉鎖の原則より



関連フォルダ　Scenes/PlayScene/Enemy

構造


更新処理を呼び出す
処理と遷移条件
を記述
クライアント
EnemyObject
ステート.1
Enemy_IdlingState
コンテキスト
EnemyStateContext
インターフェース
IEnemyState
現在のステートを
再生する

カスタムシェーダー

ユーザビリティの向上

アウトライン

プレイヤーがアクションを起こせるオブジェクトが
分かりにくかったため実装　　実装の詳細は20P~

シルエット

ユニットが他の3Dオブジェクトの背後に
隠れてしまった場合の視覚補助として実装

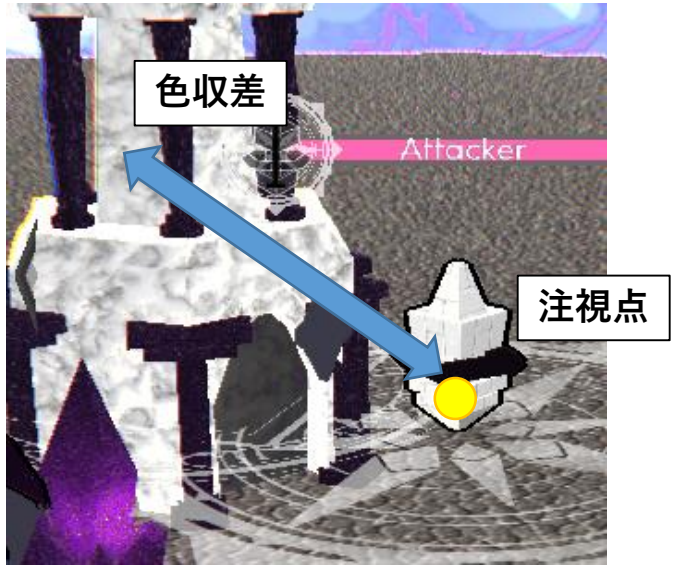
シルエット用に深度ステンシルバッファを
変更したモデル描画と
通常描画の二回ドローコールを行い実現

アルケミフォートレス

カスタムシェーダー

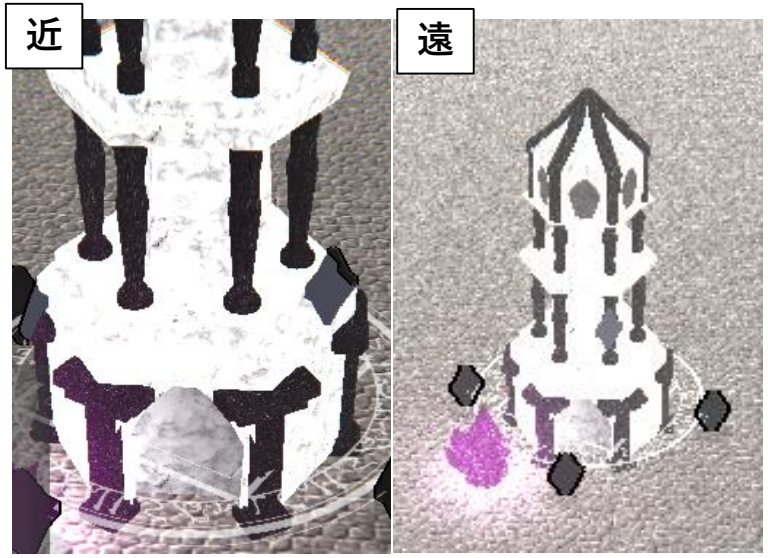
空気感の演出

色収差



注視点から一定距離
離れた位置に適応

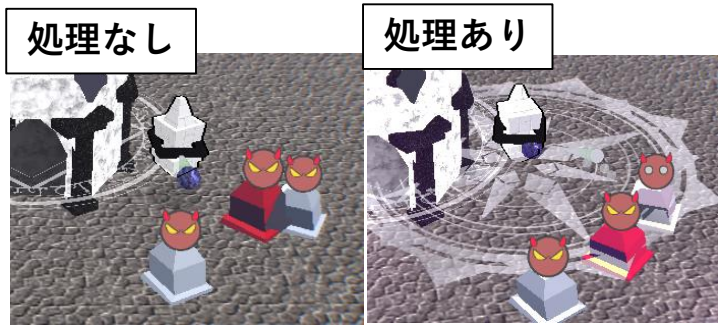
距離フォグ



カメラ位置からの距離を参照

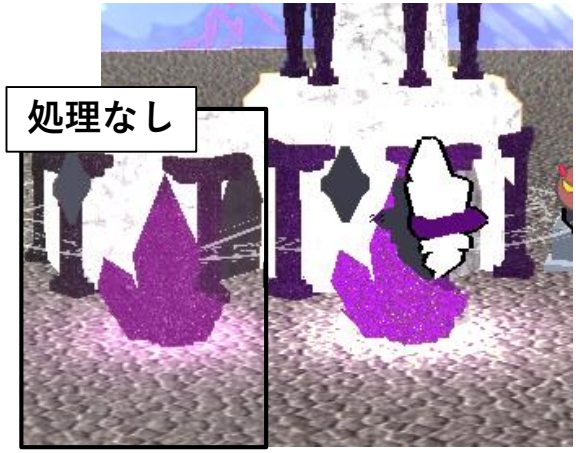
その他

オーバーレイ,乗算を用いた色味の調整



鮮明にさせない ふんわりとした雰囲気に

ノーマルマップの影響を受けるポイントライト



得られる効果

余韻のある画面作り

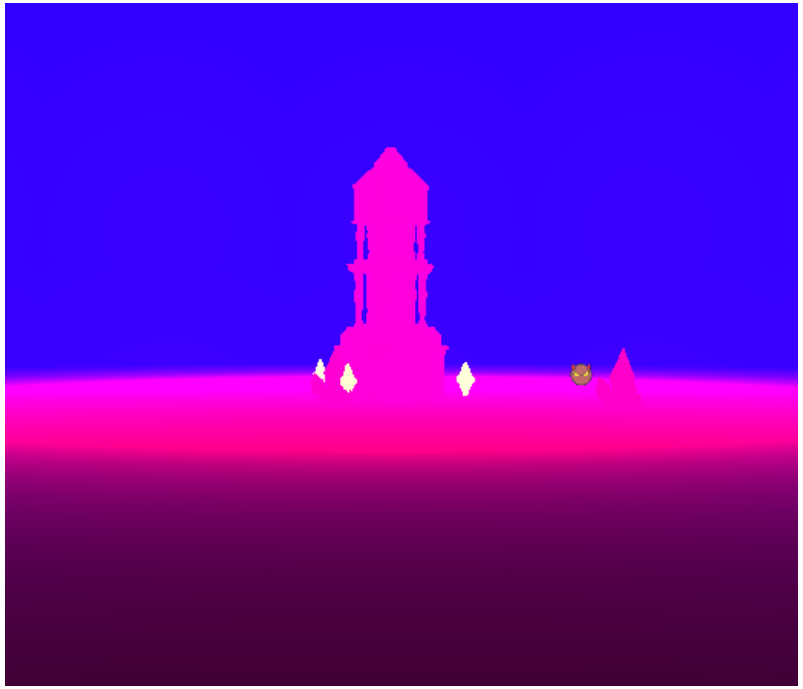
注視点の明確化

遠近感のある画面作り

カスタムシェーダー

マルチパスレンダリング

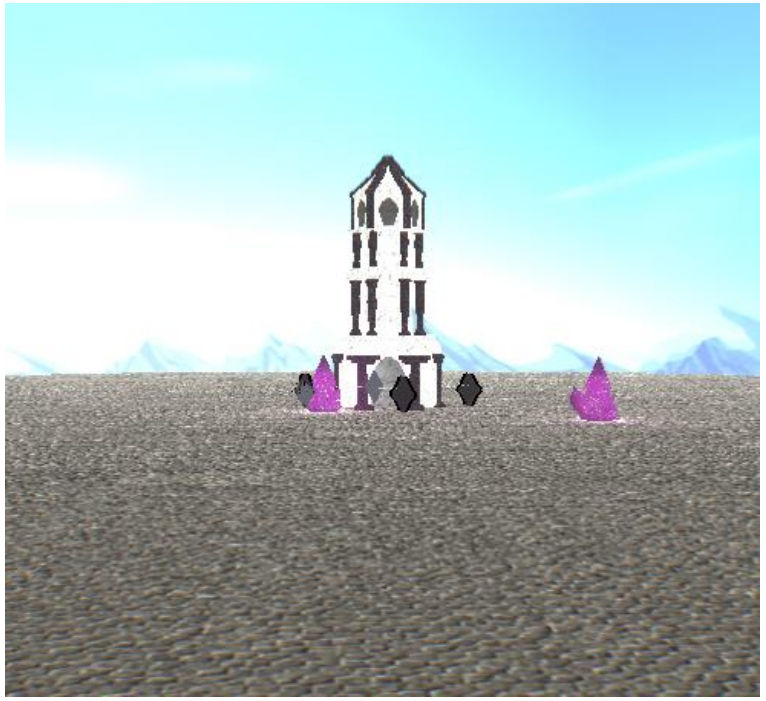
1パス目



R：注視点からの影響距離
G：アウトラインオブジェクトのみ着色
B：カメラからの距離

Gバッファに書き込む情報

2パス目



情報を元に制作した効果

色収差
アウトライン
距離フォグ

関連ファイル　Resource/Shader/PostProcess_PS.hlsl

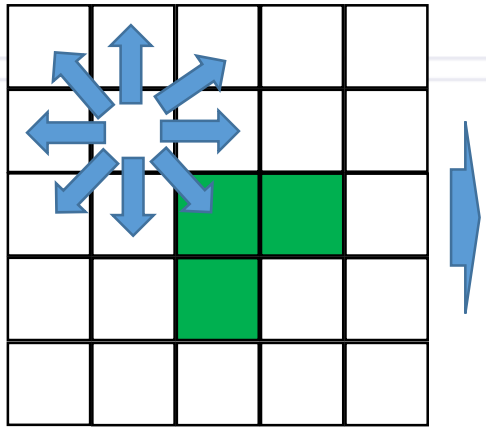
カスタムシェーダー

アウトライン

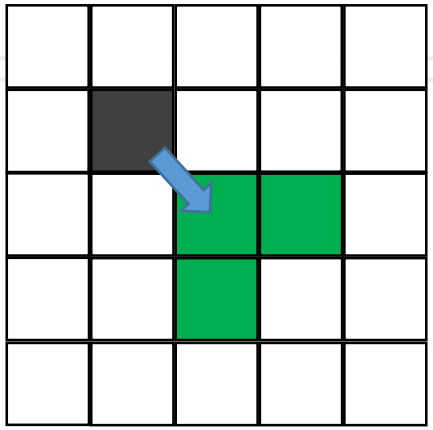
採用理由

本作にて多用するハードエッジモデルに対応したアウトラインを生成できる
モデル側の法線を変更する必要が無い

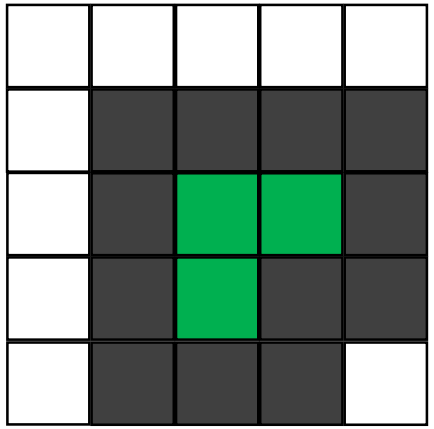
```
float AddOutLine(float power, float2 uv, float offset)
{
    float lineflag = 0.0f;

    lineflag -= 8.0f * step(offset, dTex.Sample(samLinear, float2(uv.x, uv.y)).g);

    // 八方向サンプリング（厳密には9回）
    for (int i = 0; i < 9; i++)
    {
        // サンプリング位置から指定方向のG成分を受け取る
        float judgement = dTex.Sample(samLinear, uv + float2((((float) i % 3) - 1), (((float)i / 3) -1)) * power).g;

        // 満たしていたら加算
        lineflag += step(offset, judgement);
    }

    return lineflag;
}
```



サンプリング位置から
八方向にG成分がある
かを調べる



一つでもあれば
アウトラインとする
サンプリング位置にG成分が
あれば書き込まないようにする



結果、G成分の周辺がア
ウトラインになる

さいごに

下記リンクは
これまでに制作したゲームをまとめた
ポートフォリオサイトになります

https://sites.google.com/trident.ac.jp/nagaseportfolio/%E3%83%9B%E3%83%BC%E3%83%A0

選考のほどよろしくお願いします