作品概要

タイトル：「Desert Battle」

ジャンル：TPSアクション

開発期間：3週間

開発環境：DirectX11, Visual Studio 2019

開発言語：C/C++

タイトル画面

作品概要

近距離攻撃でボスを倒すTPSアクションゲーム

ポイント

ボスの属性に応じてフィールドの様相が変わる
ボスが地面に潜ると土が盛り上がる、泳ぐ動きに合わせて波が発生するなど
敵の動きに地形が影響を受ける**地形変動に注力！**

アニメーション

階層アニメーションにおける
アニメーションとアニメーションの
切り替え時にブレンドを行い
アニメーション同士の切り替えを滑らかにしています。

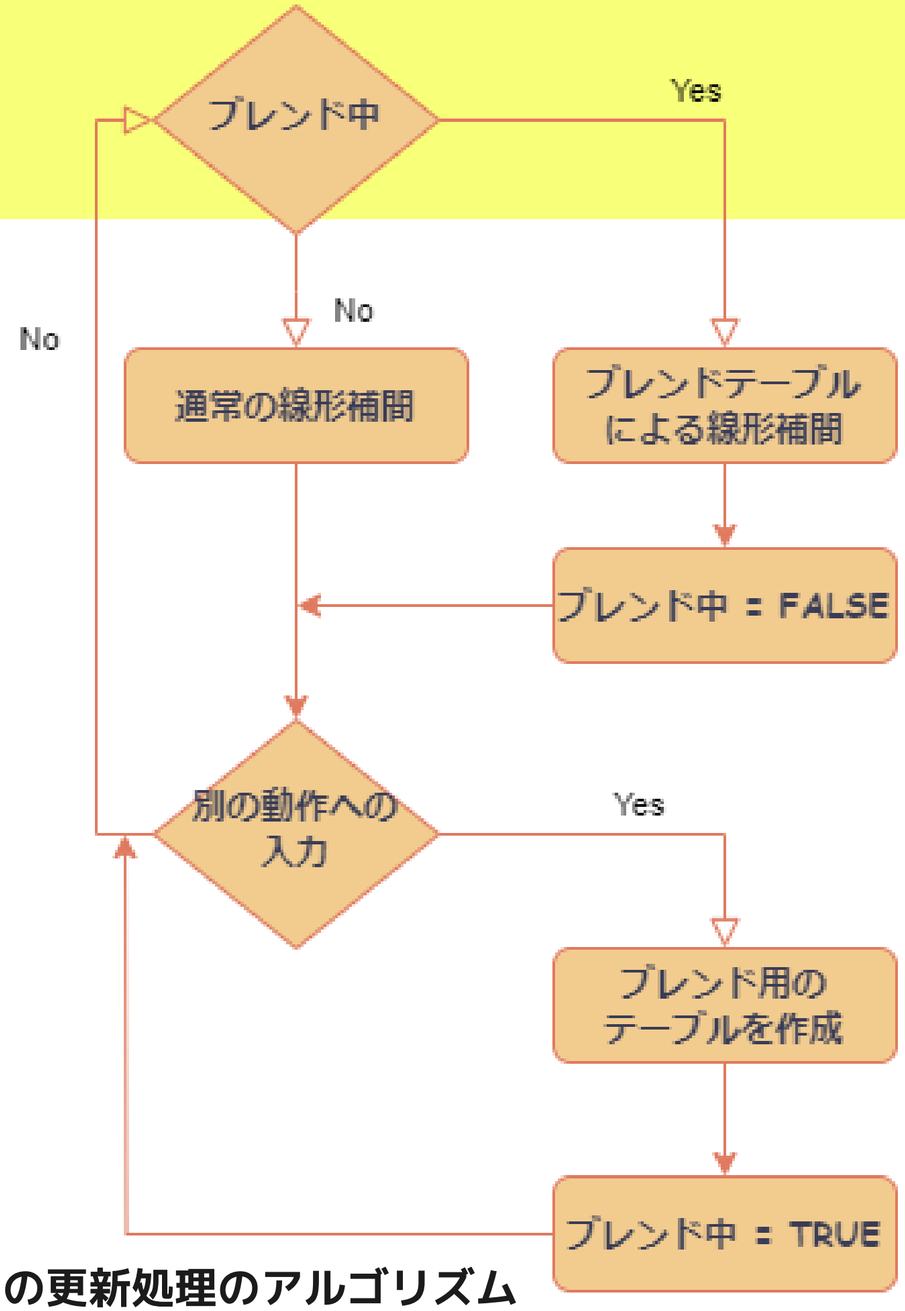
通常アニメーションの線形補間用データが
格納されたテーブルの他に、
ブレンド時用のテーブルを予め用意し処理を行っています。

アニメーション

アニメーションの更新処理

現在ブレンド中かを判断しそれによって線形補間用テーブルとして呼び出すアドレスを変更しています。

通常のアニメーションループ中に別の動作へ切り替わる入力を得ると、ブレンド用のテーブルを**その場で作成**し次のブレンドアニメーションに使用します。

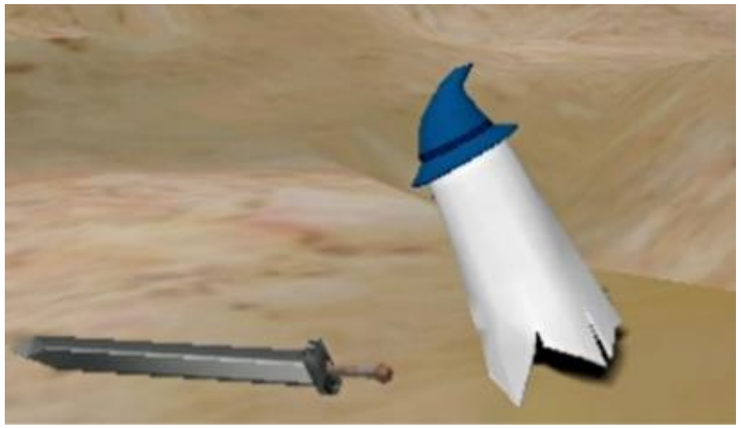


アニメーションの更新処理のアルゴリズム

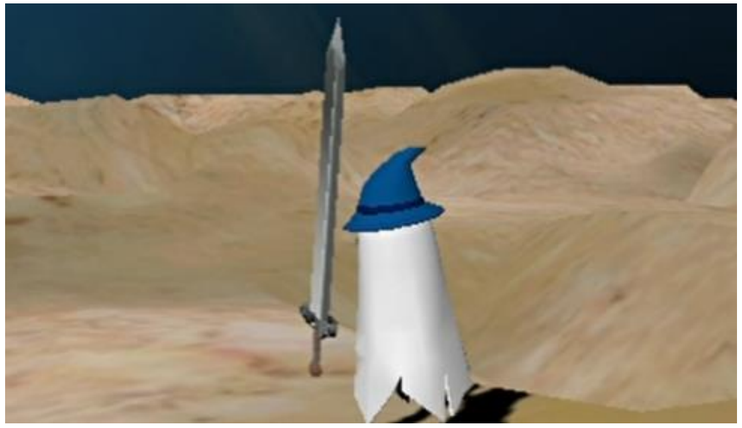
アニメーション

ブレンド用のテーブル作成詳細

ブレンド用のテーブルをClear()



各パーツにおける現在のアニメーションの最後のフレーム情報

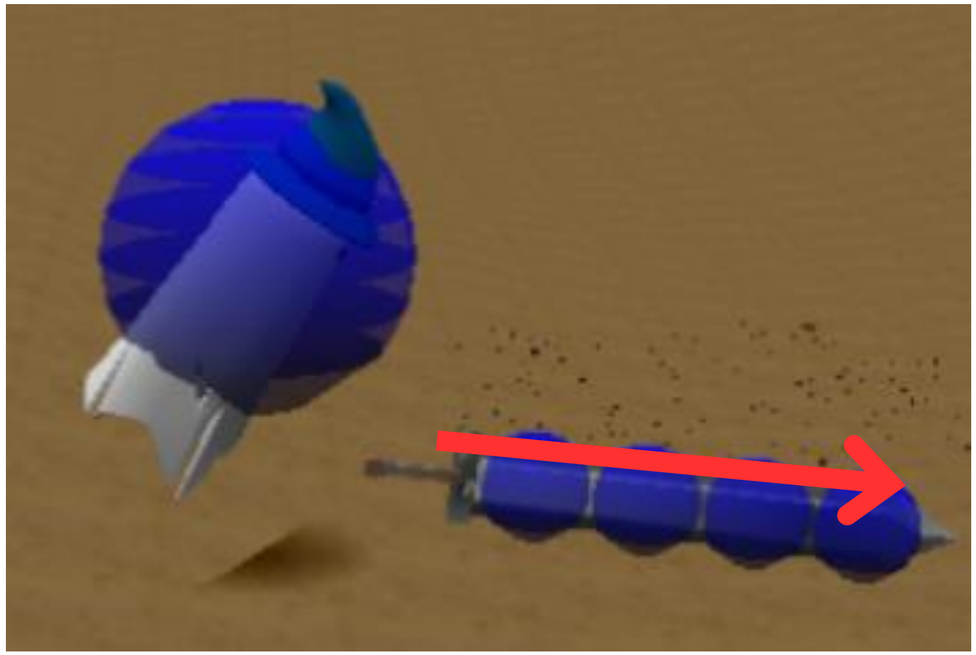


各パーツにおける次のアニメーションの最初のフレーム情報

ブレンド用のテーブルに格納

アニメーション

アニメーション中の当たり判定



当たり判定はパーツごとに付与し
デバッグモードでは可視化しています。

大剣は、
剣の持ち手から剣先へのベクトルに対し
BC(バウンディングサークル)を複数個並べ
それぞれに対して判定を行っています。

アニメーション

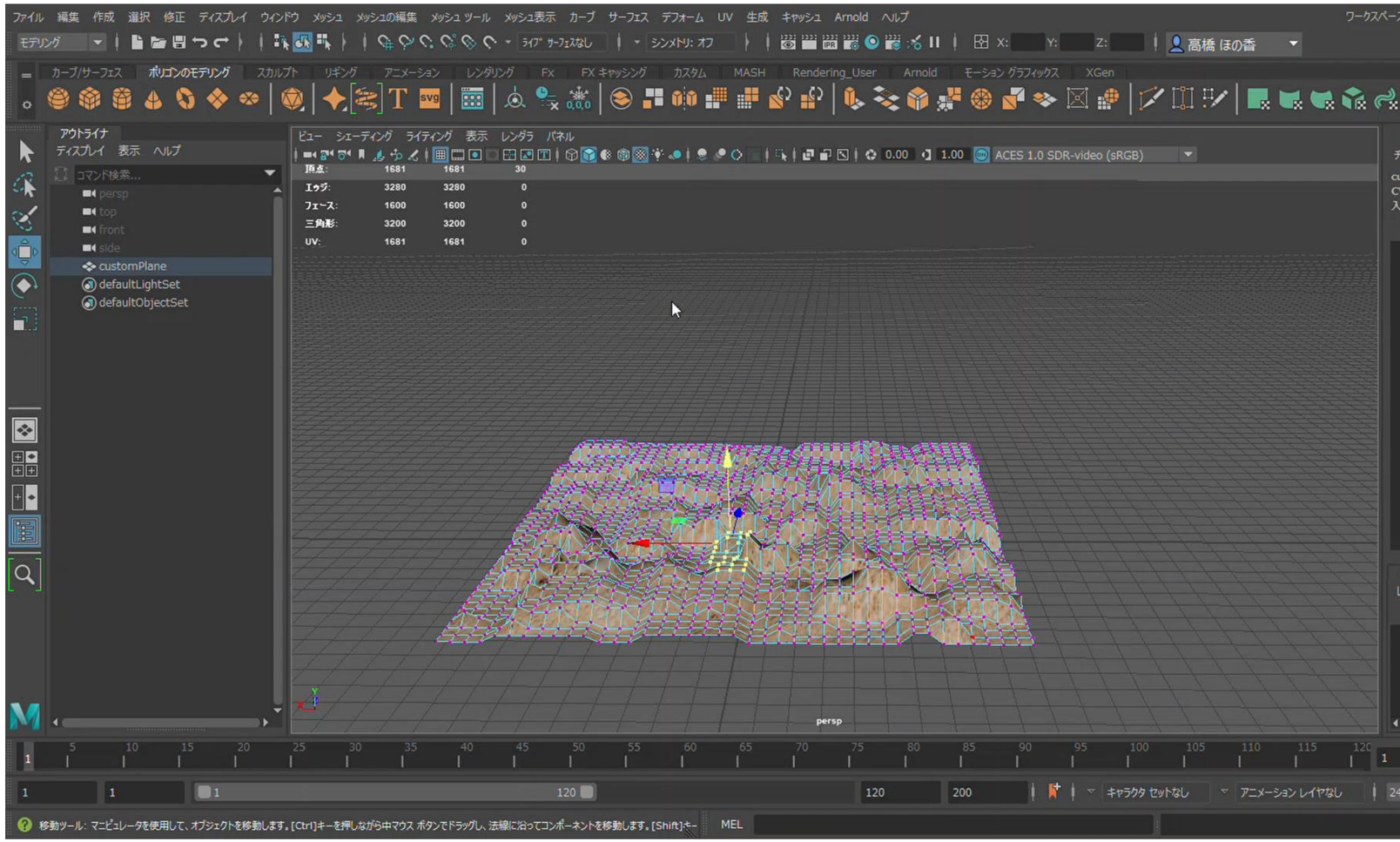
アニメーション中の当たり判定

・今後の課題
モンスターのモデルは関節が多いので
階層アニメーションの動きでは限界がある
↓
スキンメッシュアニメーションにも
挑戦したい

フィールドエディター

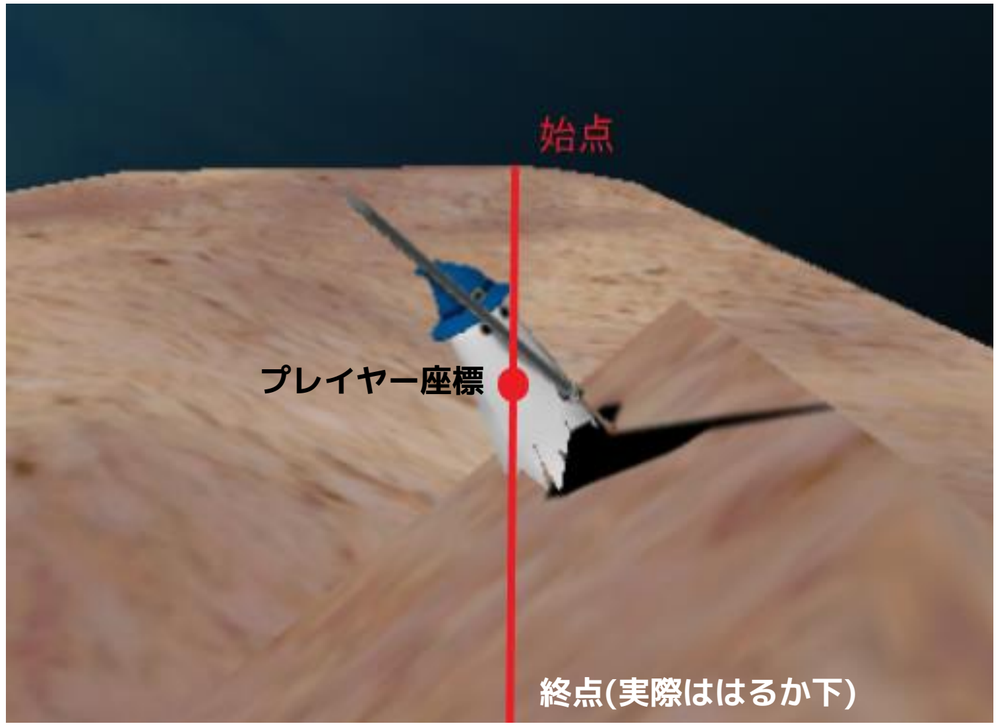
Pythonを使ってMAYA上で動作するフィールドエディターを作成しました。

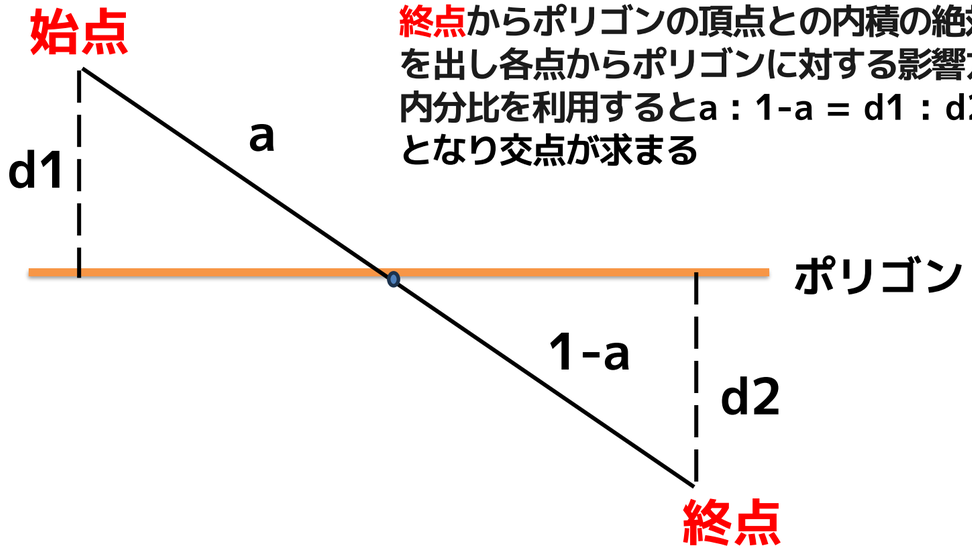
任意の数字を入力し、その数字×数字のマスで構成された正方形の板ポリゴンを作成
↓
各頂点座標を移動させ地形を編集するほか、MAYA上のペイントツールで作成したテクスチャを使用することでゲームでも同じ様相に



フィールドエディター

ゲームではプレイヤーの頭上から真下に向かって
レイキャストを行い衝突しているポリゴンとの
交点を求め地面との当たり判定とするため、
エディター上で持ち上げた頂点には登ることが可能です。





始点からポリゴンの頂点との内積の絶対値(d1)
終点からポリゴンの頂点との内積の絶対値(d2)
を出し各点からポリゴンに対する影響力の大きさを求め
内分比を利用するとa：1-a = d1：d2
となり交点が求まる

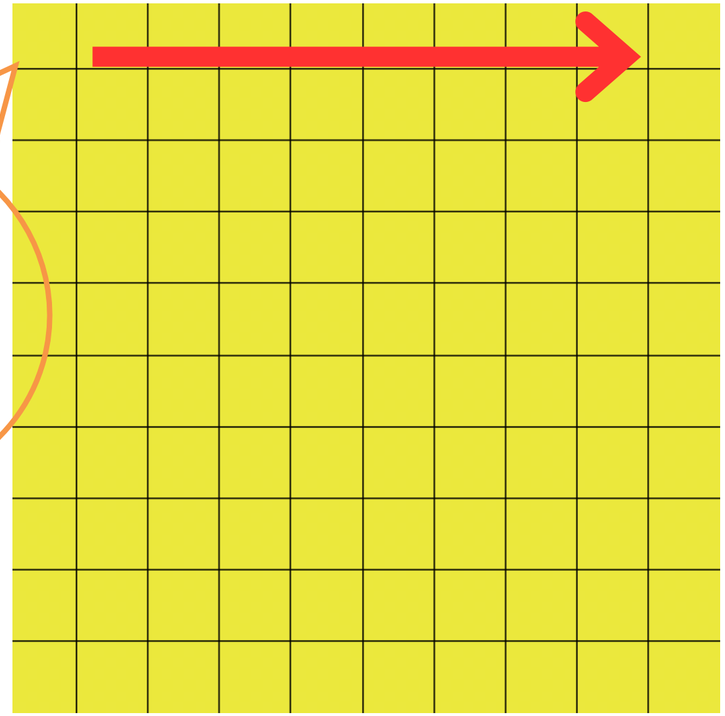
フィールドエディター

苦労したこと

MAYAとゲームで使う板ポリゴンの頂点配列の出力順の違い

例: 10×10マスの場合

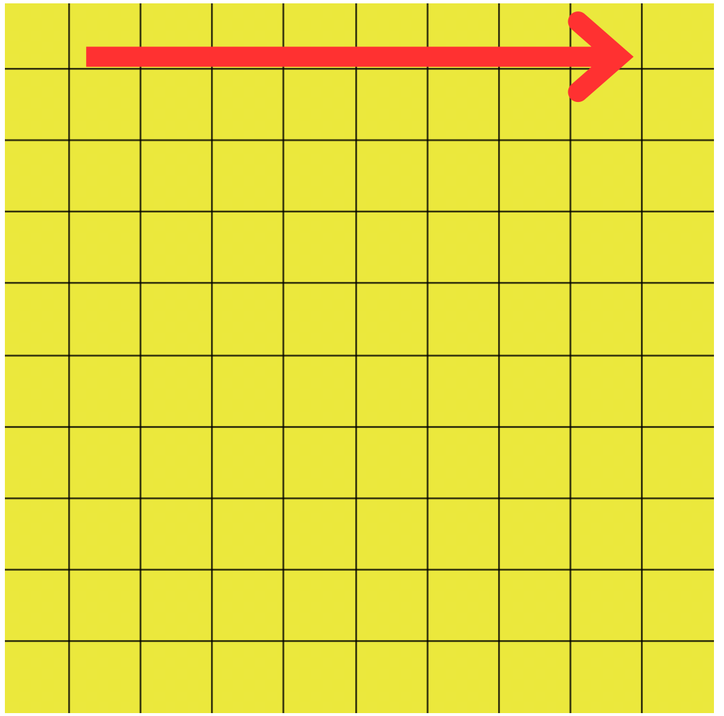
0 11 22 33 44 55 66 77 88 99 110



そのまま出力してもゲームではMAYAの見た目通りにならない

MAYA

0 1 2 3 4 5 6 7 8 9 10



ゲーム側のメッシュフィールド

フィールドエディター

対策：
頂点数に応じて配列をMAYA側のパターンに
並べ直してから出力するよう実装

↓

ゲーム側ではInitで頂点を順にループする際に
MAYAから出力された情報を上から順に読み
込めばよい
ゲーム側での導入が容易に！

☆MAYA側の配列パターンを再現
与えられた頂点数が平方数か判定
→平方数であれば平方根を取得
x軸の順番(通常)×平方根+y軸の順番(通常)
という法則で0~頂点数の数字を埋めていく

```
#------------------------------------------------------------
# パターンを計算
#------------------------------------------------------------
def CalculatePattern(num_vertices):
    if int(num_vertices**0.5)**2 != num_vertices:
        print("指定された数字は二乗数ではありません。")
        return None

    pattern = []
    sqrt_num = int(num_vertices**0.5)

    for i in range(sqrt_num):
        for j in range(sqrt_num):
            value = i + j * sqrt_num
            if value not in pattern:
                pattern.append(value)

    return pattern
```