

ガンプラ大好きプログラマーのポートフォリオ

名前： 永松 涼華

学校： 福岡情報ITクリエイター専門学校

趣味： プラモデル制作, 読書

使用可能ツール/スキル

C++	2年
C#	1年
Unity	1年
GitHub	2年
Adobe PhotoShop	3年
Blender	半年
Word/Excel/PowerPoint	8年



自己PR

・皆勤

無遅刻

無欠席

無早退

全て達成中です！！



・プラモデル制作

1年半ほど前から始めました。
最近は、ガンダムマーカーを
使って少しでも塗装に挑戦し
ています。



無遅刻無欠席三人集
です！持っている紙
には書いていません
が、無早退です！

現在はこのような感
じで飾っています。
これからも増えてい
くので棚の増設を検
討中。

最終制作作品の紹介



最終制作作品

作品名 : AscensioQuest

ジャンル : 3Dアクションゲーム

制作環境 : C++/DxLib

制作期間: 2024/10/1~2025/3/1

制作人数: 1人(個人制作)

担当箇所: プログラムとUI作成

ゲーム内容:

襲い来る敵を切り倒そう！！

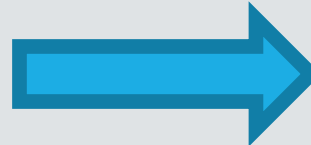
GitHub URL <https://github.com/Remy-1216/AscensioQuest>



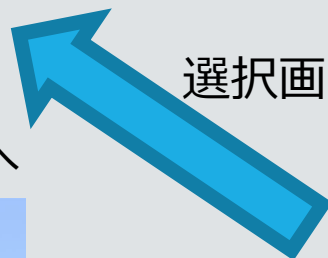
ゲームフロー



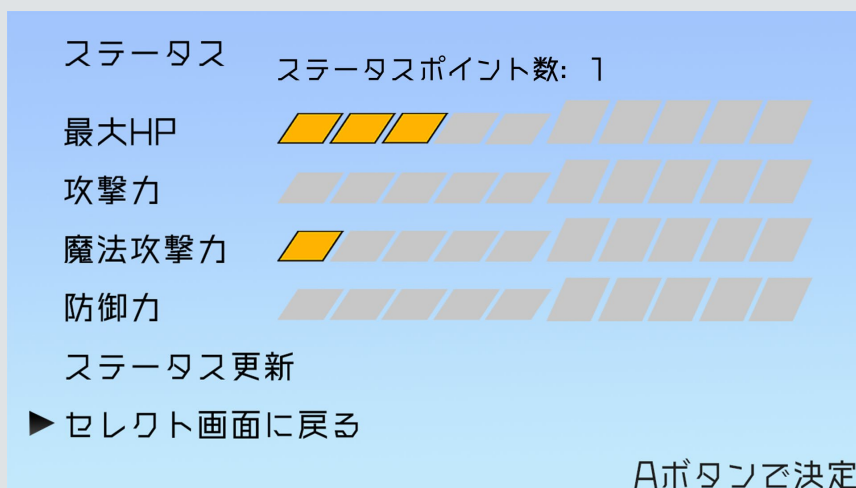
ステージ選択



選択画面に戻る



敵を全員倒す



Aボタンで決定

GameClear

Aボタンを押すと、セレクト画面に戻る
Bボタンを押すと、ゲーム終了

ここが面白い！①

■ 様々な**攻撃**が可能！！

剣での攻撃に加え、魔法攻撃などで敵を倒していくことができます！
敵に合わせて攻撃方法を変えていきましょう！

剣で攻撃！！



魔法で遠距離攻撃！！




ここが面白い！②


■ 自分好みに**ステータス**を強化できる


敵を倒すことで、獲得したステータスポイントを使用すると
ステータスを強化することができます！


自分の好みにステータスを上げていきましょう！！

ステータス ステータスポイント数: 10

▶ 最大HP 

攻撃力 

魔法攻撃力 

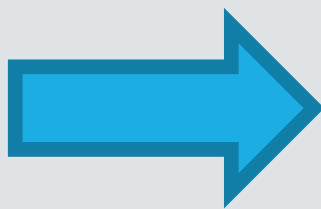
防御力 

ステータス更新


セレクト画面に戻る


Aボタンで能力を上げる


自分好みに**強化**！




ステータス ステータスポイント数: 0

最大HP 

攻撃力 

魔法攻撃力 

▶ 防御力 

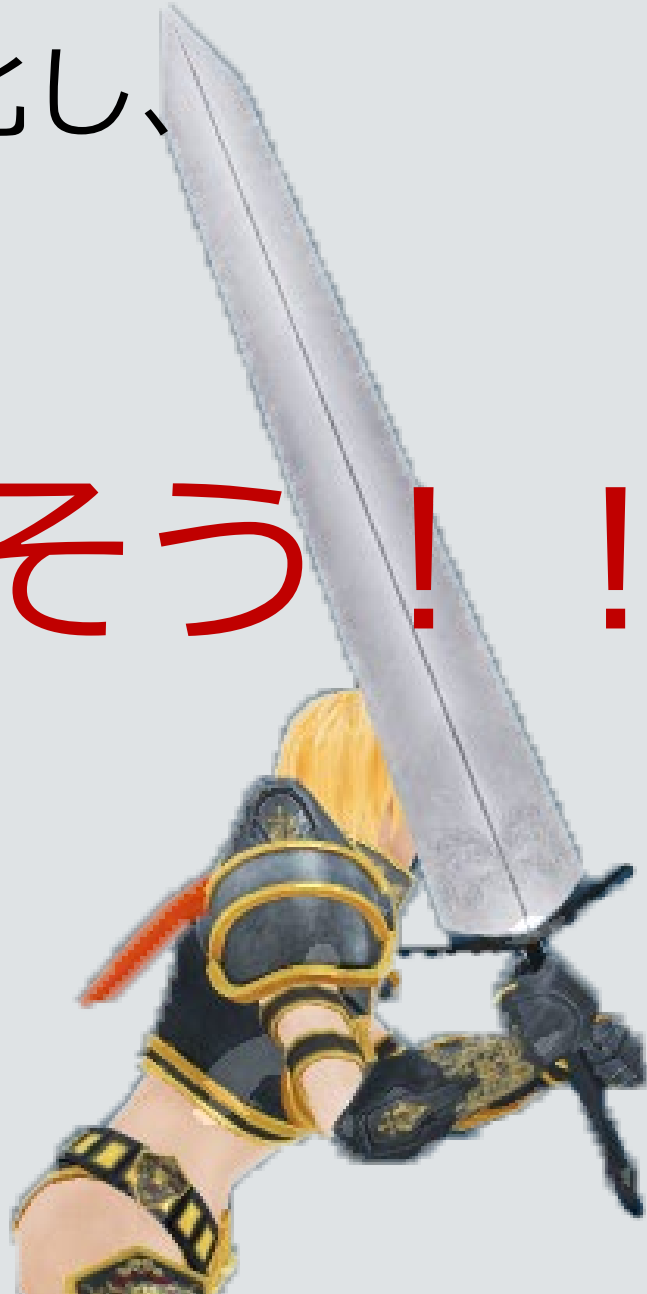
ステータス更新

セレクト画面に戻る

Aボタンで能力を上げる

ステータスを強化し、

ボスを倒そう！！



技術紹介



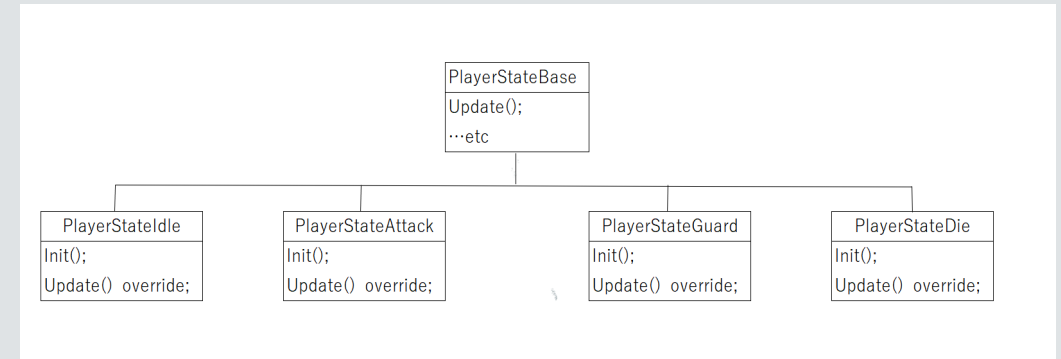
■ 技術紹介① ステートパターン

プレイヤーと敵の状態管理
をステートパターンで実装しました。

その結果！！

- ・ 各状態事の**処理**を行いやすくなった
- ・ バグが起きた時、
すぐに**解決**できるようになった！！

↓ クラス図



↓ PlayerStateAttack.cpp 18行目

```
void PlayerStateAttack::Update(Stage& stage, const Pad& pad, const Camera& camera)
{
    //アニメーションのスピードを受け取る
    m_animTime = m_pPlayer->GetAnimSpeed();

    //ステージとの当たり判定
    m_pPlayer->Move(stage, VGet(0.0f, 0.0f, 0.0f));

    //アニメーションがどれだけ進んでいるか
    m_time += m_animTime;

    // 攻撃処理
    if (pad.IsTrigger("A"))
    {
        m_aButtonCount++; // Aボタンを押した回数を記録
    }

    //攻撃の処理
    Attack();

    // 状態遷移
    // 攻撃から待機状態に変更
    if (m_pPlayer->GetAnimLoopEndTime() <= m_time)
    {
        m_nextState = std::make_shared<PlayerStateIdle>(m_pPlayer);
        auto state = std::dynamic_pointer_cast<PlayerStateIdle>(m_nextState);
        state->Init();
    }
}
```

■ 技術紹介② 外部ファイル化

プレイヤーや敵の当たり判定の大きさや
プレイヤーや敵のステータスのデータ
などをcsvファイルで管理しました。

その結果！！

- 値の**調整**など素早くできるようになった
- キャラクターの**追加**が行いやすくなった

↓LoadCSV.cpp 48行目

```
//キャラクターのステータスを読み込む
void LoadCsv::LoadStatus(CharacterBase::Status& data, const char* charaName)
{
    std::ifstream ifs(kCharaStatusFileName);
    std::string line;
    std::vector<std::string> strvec;
    m_data.clear();

    while (std::getline(ifs, line))
    {
        strvec = split(line, ',');
        const char* str = strvec[0].c_str();

        // 参照したいキャラが見つかったら処理をやめる
        // MEMO:strcmp 文字列を比較する 第1引数 = 第2引数の場合0
        if (strcmp(str, charaName) == 0)
        {
            break;
        }
        else
        {
            strvec.erase(strvec.begin(), strvec.end());
        }

        // ステータス情報を代入する
        data.maxHp = std::stof(strvec[1]);
        data.maxMp = std::stof(strvec[2]);
        data.attackPower = std::stof(strvec[3]);
        data.magicAttackPower = std::stof(strvec[4]);
        data.defensePower = std::stof(strvec[5]);
        data.walkSpeed = std::stof(strvec[6]);
        data.runSpeed = std::stof(strvec[7]);
    }
}
```

↓キャラクターのステータスデータ

ID	HP	MP	攻撃力	魔法攻撃力	防御力	歩く時の速さ	走るときの速さ
player	100	100	15	50	0	6.0f	12.0f
shortdista	50	0	5	0	0	2.0 f	0.0 f
longdistan	50	0	5	0	0	2.0 f	0.0 f
boss	450	0	20	0	0	2.0f	0.0f
tutorialene	100	0	10	0	0	2.0f	0.0f

■ 技術紹介③ ボスの挙動

ボスには様々な行動を行ってもらいたかったため、ボスのHP量とプレイヤーの距離で行動を変更するようなAIの実装を行いました。

その結果！！

- ・ ボスが**状況**に応じて行動するようになった
- ・ ボスが**様々な攻撃**を行うようになった

↓BossAI.cpp 47行目

```
//どのような状況下か
int BossAI::StateSet(CharacterBase& boss,const Player& player)
{
    //敵とプレイヤーの距離
    m_distance = VSize(VSub(boss.GetPos(), player.GetPos()));

    if (boss.GetHp() >= kBossHp && m_distance <= kShortDistance)
    {
        LoadCsv::GetInstance().LoadBossAIData(m_bossAI, kHpHighlyRangeNear);
        return AIMotion(m_bossAI);
    }
    if (boss.GetHp() >= kBossHp && m_distance >= kLongDistance)
    {
        LoadCsv::GetInstance().LoadBossAIData(m_bossAI, kHpHighlyRangeFar);
        return AIMotion(m_bossAI);
    }
    if (boss.GetHp() <= kBossHp && m_distance <= kShortDistance)
    {
        LoadCsv::GetInstance().LoadBossAIData(m_bossAI, kHpLowRangeNear);
        return AIMotion(m_bossAI);
    }
    if (boss.GetHp() <= kBossHp && m_distance >= kLongDistance)
    {
        LoadCsv::GetInstance().LoadBossAIData(m_bossAI, kHpLowRangeFar);
        return AIMotion(m_bossAI);
    }

    return kWalk;
}
```

↓ボスがどのような行動を行うかの確率をCSVに保存し、読み込んでいます

	kLightAttack	kStrongAttack	kThrowing	kStatusUp
HpHighlyFar	70	80	0	100
HpHighlyNear	0	0	80	100
HpLowFar	80	70	0	100
HpLowNear	0	0	70	100

■ 技術紹介④ 当たり判定

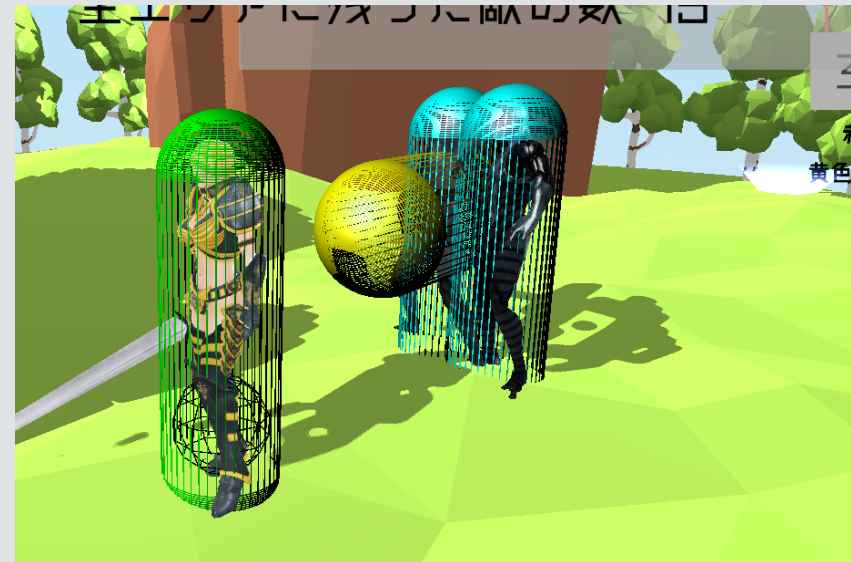
当たり判定を細かく行うために胴体と攻撃など細かく当たり判定を設定し実装を行いました。

その結果！

- ・ 攻撃を受けたときのタイミングにダメージを受けた**アニメーション**や**エフェクト**が表示できるようになった。



↓攻撃の当たり判定(黄色のキューブ)



作品総括

AI制御をしっかりとする！！

現在の敵AIでは、偏った行動を行うことがあります。
今回は試験的に自分の考えたAIだったので、書籍などを活用し
敵AIについて学んでいき、実装していきます。

そのほかには、UIの改良なども行っていきたいと考えています。



魔族の森



過去作品の紹介



Monster
Crusher



過去作品 その1

作品名: MonsterCrusher

ジャンル: 3Dアクションゲーム

制作環境: C++/DxLib

制作期間: 2024/6/1~2024/9/1

制作人数: 1 人(個人制作)

GitHub URL: <https://github.com/Remy-1216/MonsterCrusher>



過去作品 その2

作品名: 魔族の森

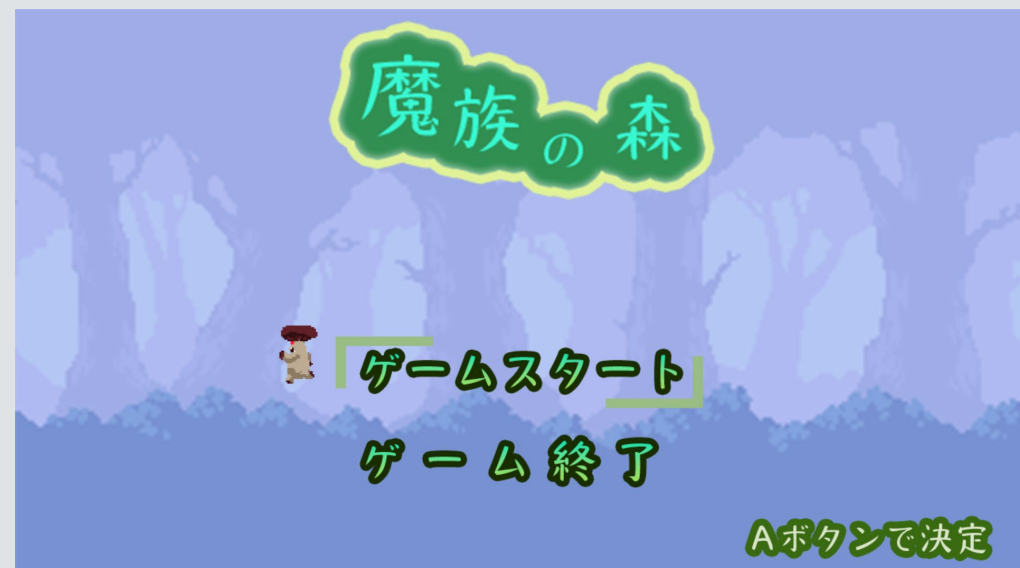
ジャンル: 2Dゲーム(避けゲー)

制作環境: C++/DxLib

制作期間: 2023/10/1~2024/2/15

制作人数: 1人(個人制作)

GitHub URL : <https://github.com/Remy-1216/Mazokunomori>



今後の目標

敵AIについて勉強していく！

今回は試験的な自分なりの考えで制作したAIでしたが、
しっかりとした敵AIについての仕組みを学びたいと
思っています。

敵AIについて学び、ボス以外のモブ敵などにも、
それぞれのAIを実装した作品を制作したいと思っています。