新進気鋭なグラフィックプログラマーのポートフォリオ

名前　西村　龍太
学校　福岡情報ITクリエイター専門学校
趣味　バスケ、外食
GithubURL https://github.com/niyugitHub
スキル：C++　２年
C#　１年
HLSL　１年
Git　1年半
Unity　１年

最終制作作品

作品名	アタックフィーバー
ジャンル	3Dアクションゲーム
開発環境	C++,HLSL,DxLib
対応機種	Windows
制作期間	５か月
制作時期	2023年9月~2024年1月
担当	モデル以外すべて

アタックフィーバー　企画

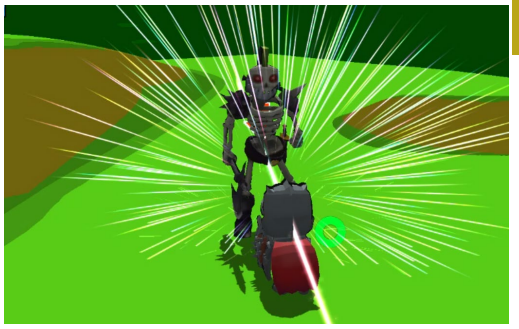
ジャンル
3Dアクションゲーム

最終目標は町の周辺を荒らす
モンスターのボスを倒すこと

次々と出てくるモンスターから
様々な攻撃を用いて戦う

←通常攻撃

回転切り→



←必殺技

アタックフィーバー　HLSL

HLSLのシェーダーを使って、以下の実装を行った

攻撃を食らったときプレイヤーを　赤くする →

モデルの輪郭線
影（シャドウマップ）→
トゥーンシェーダー

敵が死亡した際モデルを徐々に黒くする →

アタックフィーバー　当たり判定

フィールドのポリゴンの法線の向きによって壁と床に分類し、当たり判定の処理を行った

壁に当たった時の処理

```
//壁を考慮した移動を外積を使って算出
VECTOR SlideVec;

// 進行方向ベクトルと壁ポリゴンの法線ベクトルに垂直なベクトルを算出
SlideVec = VCross( In1: dynamic_cast<CharacterBase*>(this)->GetInfo().vec, In2: m_pPoly->Normal);

// 算出したベクトルと壁ポリゴンの法線ベクトルに垂直なベクトルを算出、これが
// 元の移動成分から壁方向の移動成分を抜いたベクトル
SlideVec = VCross( In1: m_pPoly->Normal, In2: SlideVec);

// それを移動前の座標に足したものを新たな座標とする
m_nextPos = VAdd( In1: m_oldPos, In2: SlideVec);
```

床に当たった時の処理

```
//一番高い床ポリゴンにぶつける為の判定用変数を初期化
float PolyMaxPosY = 0.0f;

// 床ポリゴンに当たったかどうかのフラグを倒しておく
m_isHitFlag = false;

// 床ポリゴンの数だけ繰り返し
for (int i = 0; i < m_floorNum; i++)
{
    // i番目の床ポリゴンのアドレスを床ポリゴンポインタ配列から取得
    m_pPoly = m_pFloorPoly[i];

    // 頭から足元まででポリゴンと当たっているかを判定
    m_lineRes = HitCheck_Line_Triangle( LinePos1: VAdd( In1: m_nextPos, In2: VGet( x: 0.0f, y: kHeadHeight, z: 0.0f)),
        LinePos2: m_nextPos, TrianglePos1: m_pPoly->Position[0], TrianglePos2: m_pPoly->Position[1], TrianglePos3: m_pPoly->Position[2]);

    // 当たっていなかったら何もしない
    if (!m_lineRes.HitFlag) continue;

    // 既に当たったポリゴンがあり、且つ今まで検出した床ポリゴンより低い場合は何もしない
    if (m_isHitFlag && PolyMaxPosY > m_lineRes.Position.y) continue;

    // ポリゴンに当たったフラグを立てる
    m_isHitFlag = true;

    // 接触したＹ座標を保存する
    PolyMaxPosY = m_lineRes.Position.y;
}
```

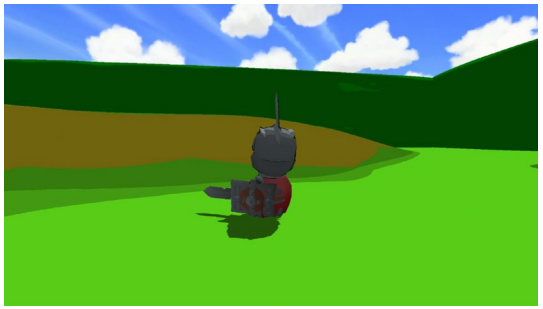
アタックフィーバー　カメラワーク

ただ辺りを見渡せるだけではなく、
自分の向いている方向と敵の
位置に合わせて最適な敵を
ロックオンする仕組みを実装

ロックオン時には敵と自分が
少し斜めの位置関係に
なるよう調整している

アタックフィーバー　カメラワーク

ゲームクリア時にカメラが
プレイヤーの周りを回りながら
ゲームクリアの演出が出る処理を
行列とイージングを使って実装



アタックフィーバー　コンボ

ゲームの面白さをより出すために
コンボ機能を実装

敵に攻撃を当てるたびにコンボが増えていき
コンボゲージが切れるもしくは敵から
攻撃を食らうことでコンボが途切れる

コンボ数を増やしていくとコンボのUIの
色が変わり、ユーザーもよりコンボを
増やしたいという気持ちになる

アタックフィーバー　駆け引き

敵によってステータスと近接攻撃、遠距離攻撃
で分け、駆け引きが単調にならないように調整

近距離攻撃 ⟶

各ボスには基本4種類の攻撃を実装し、
プレイヤーとの距離によって攻撃の種類を変え
ユーザーに駆け引きの楽しさを味わってもらう
ために工夫した

遠距離攻撃 ⟶

アタックフィーバー　駆け引き

ボス敵の攻撃の
バリエーション

通常攻撃

ショット攻撃

タックル攻撃

隕石を降らせる

アタックフィーバー　その他

Singleton,Observer,Stateパターンを使用してプログラムの
可読性を上げることができた

```
void ButtonManager::AddButton(std::function<void(void)> listener, ButtonIndex index, Vec2 center,
    int width, int height, std::vector<const char*> text, int graphHandle, int stringNum, Font::Id id, ButtonType type)
{
    //ボタンの番号
    int buttonNum = static_cast<int>(m_pButton.size());

    //ボタンの追加
    m_pButton.push_back( _Val: std::make_shared<Button>( _Args: buttonNum, _Args: index, _Args: center, _Args: width, _Args: height,
        _Args: text, _Args: stringNum, _Args: graphHandle, _Args: Font::GetInstance().GetFontData(id), _Args: type));

    //ボタンを押されたときの処理を登録
    m_pButton.back()->AttachListener(listener);
    m_pButton.back()->SetUiManager( _pButtonManager: this);
}
```

プレイヤーの
Stateパターン
↓

```
class PlayerState
{
public:
    //プレイヤーの状態
    enum class StateKind
    {
        Idle,               //待機
        Walk,               //歩き
        Dash,               //ダッシュ
        Jump,               //ジャンプ
        KnockBack,          //ノックバック
        Attack,             //攻撃
        SkillAttack,        //スキル攻撃
        SpecialAttack,      //必殺技
        Gaude,              //ガード
        JumpAttack,         //ジャンプ攻撃
        Dead,               //死亡
        StateClear,         //ステージクリア
    };
```

エフェクトの管理で使った
Singletonクラス
↓

```
public:
    ~Effekseer3DManager();

    /// <summary>
    /// Effekseer3DManagerはGetInstance()を通した参照からしか利用できない
    /// </summary>
    /// <returns></returns>
    static Effekseer3DManager& GetInstance()
    {
        static Effekseer3DManager instance;

        return instance;
    };
```

↑
ボタン本体に機能を持たせる
ためのObserver

制作実績１　2023年8月(1か月)

作品名　幽霊館

ジャンル　3Dホラー

開発環境　C#,Unity

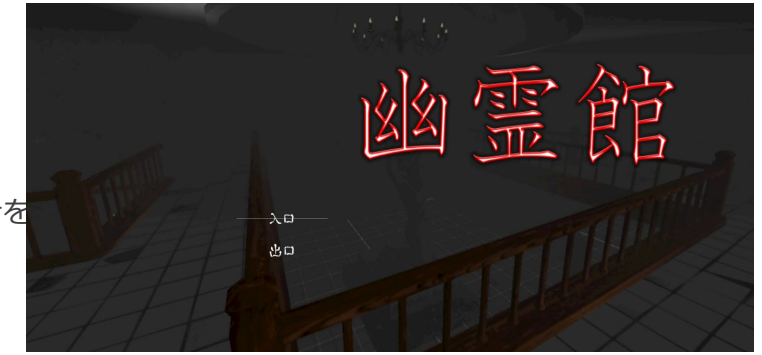
対応機種　Windows

制作期間　1か月半

制作時期　2023年7月中旬～2023年8月

担当　プレイヤー、カメラ、ミニマップ、襲われた際の演出

頑張ったこと、学んだこと
・チーム制作経験
・初めてのUnityを用いた制作
・ミニマップの作成
・Character Controller Componentを使ったプレイヤーの挙動
・カメラの挙動
・襲われた際の演出



制作実績２　2023年6月~7月(2か月)

作品名　BlockJumper

ジャンル　3Dパズルアクション

開発環境　C++,DxLib

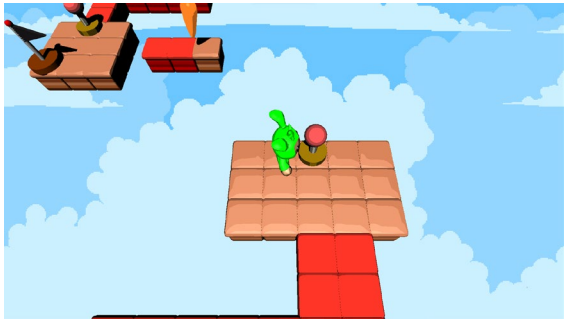
対応機種　Windows

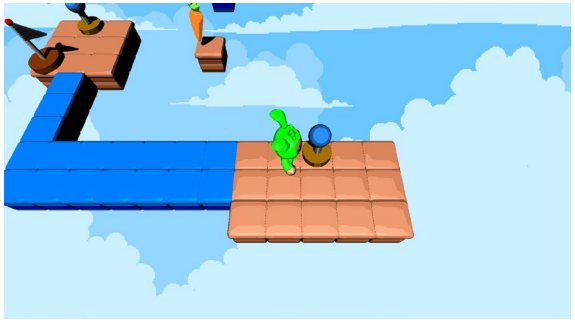
制作期間　2か月

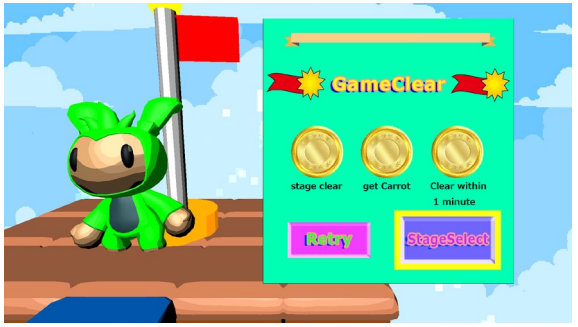
制作時期　2023年5月中旬～2023年7月中旬

担当　モデル以外全て

頑張ったこと、学んだこと
・初めての3Dゲーム制作
・カメラワーク
・セーブデータの作成







制作実績３　2023年4月~5月(2か月)

作品名　MFM

ジャンル　2D格闘ゲーム

開発環境　C++,DxLib

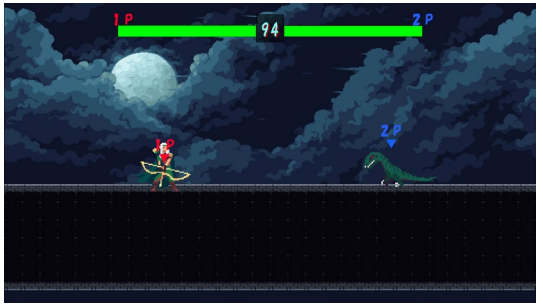
対応機種　Windows

制作期間　3か月

制作時期　2023年5月～2023年7月

担当　当たり判定、ファイター１体実装、
複数コントローラー繋げる

頑張ったこと、学んだこと
・チーム制作経験
・チームでのGitHub Desktop利用
・Boxの当たり判定
・複数のパッド処理
・Stateパターン実装



制作実績４　2023年1月~3月(2か月半)

作品名　HelloKnight

ジャンル　2Dアクション

開発環境　C++,DxLib

対応機種　Windows

制作期間　2か月半

制作時期

2023年1月～2023年3月　中旬

担当　全部

頑張ったこと、学んだこと
・Platinumというツールを用いて
マップ制作をしそれをcsvで保存
・Factory Methodパターンで
Enemyの生成管理

今後の目標(グラフィックプログラマーになるために)

グラフィックスプログラマーになりたいと考えているため参考書を読んでHLSLの勉強をし、今回出来なかったシェーダーの表現を出来るようになる

DxLibは自分でシェーダーの実装をしようとすると機能が邪魔をして相性が悪いため今後DirectX12の勉強をし、DirectX12でゲームを制作する