この作品は三校合同学内コンテストで銀賞を受賞した作品です。

◇ジャンル

TPSシューティング

◇ゲーム説明

PvEモードでは出現してくる敵をひたすら倒し撃破数を増やしていくモードです

PvPモードではコントローラーが二つ必要ですがプレイヤー同士で対戦ができるようになっています

チュートリアルモードでは一通りの操作を学べます

◇操作方法（xboxコントローラー）

左スティック：移動

右スティック：視点（AIM）移動

B：ジャンプ

X：リロード

LB+左右前いずれかの移動：ブースト

RB：弾を発射

LT,RT：リーン（覗き込み）

◇その他

□モードセレクト画面時（PvPモードのみ）

・タイトル画面時に裏コマンドとしてLTを押すとステージ１、RTを押すとステージ２が選択されます。どちらも押さなかった場合ランダムでどちらかになります

・また、PvPモード選択後に出てくるシーンではカメラを動かすことができ、地形把握をすることができます

□いずれかのゲームモード時

・BACKボタンを押すと強制的にゲームモードを終了してリザルトもしくはセレクト画面に移行します。

また、PvPモード選択後に出てくるシーンではBACKボタンを押してもシーンは変わりません。

開発期間：2020年12月～2021年1月 (2ヵ月)

使用ツール：DirectX11

動作環境：Windows(XInput対応コントローラ必須)

苦労・工夫した点：開発期間が短かったことと、他の学内コンテストが被っていたので時間をやり繰りするのに苦労しました。コンポーネント指向で作成すると効率良くコーディングできると教わったのでWEBで調べて実装することで短い期間でも効率良く開発することができました。

参考にしたもの

[【REVY】進撃の台湾鉄鬼2【重量編】 - YouTube](https://www.youtube.com/watch?v=eYQT679SN8U)

[厳選！C++ アルゴリズム実装に使える 25 の STL 機能【前編】 - Qiita](https://qiita.com/e869120/items/518297c6816adb67f9a5)

[[C++]ゲームプログラムのコンポーネント指向 - Qiita](https://qiita.com/harayuu10/items/bf6d73353efa45212200)

[DirectXで画面分割 - Qiita](https://qiita.com/akurobit/items/1619bc26010441b8008c)

[C++ テンプレート　ヘッダと実装を別にした時に起こること - Qiita](https://qiita.com/i153/items/38f9688a9c80b2cb7da7)

[演算子のオーバーロード(C++) - 超初心者向けプログラミング入門 (pc-note.net)](https://programming.pc-note.net/cpp/operator.html)

[ゲームUIデータベース|ようこそ (gameuidatabase.com)](https://www.gameuidatabase.com/)

[衝突判定編 (marupeke296.com)](http://marupeke296.com/COL_main.html)

・点と扇形の当たり判定をWEBを参考にしました。

・AABBの当たり判定と押し出し処理は授業で作成したものがベースとなっています。

・カメラの挙動は授業で作成したものをベースにしています。（揺れと遅れて移動する表現は自作です。）

アピールポイント

■ゲーム部分

・Enemyに視野を持たせ視野内に入ったプレイヤーを攻撃するようにしました。

・Enemyの移動ルートがランダム変化するAIを作りました

・対戦モードを作ってオフラインで友人と遊べるようにしました。

・操作が多いゲームですが、チュートリアルモードを作ることによってゲームプレイの敷居を下げました。

・ダメージを受けた際、カメラが揺れる演出を作り臨場感が出るようにしました。

・モードセレクト画面で選択しているモードを’小刻みに動くポリゴン’で強調表示することによって、おしゃれでかっこよくしました（ペルソナ５の真似をしています）

■プログラム部分

・従来のやり方だとオブジェクトを作成するごとにそのオブジェクトの更新関数や描画関数をメイン関数に記述する必要がありました。しかし。コンポーネント指向を導入することで、新たに作成したオブジェクトはリストに加えられ追加でメイン関数に記述することなく動作するようにしました。これにより直感的にコードを記述しやすくなりました。

・ステートパターンを取り入れたことでswitch文、if文を減らし、見やすく拡張しやすいソースコードを作成できるようになりました。可読性が向上したことによりデバッグ時にバグを発見しやすくなりました。