

チームを支えるプログラマー

名前 :井上 恋

学校 :福岡情報ITクリエイター専門学校

メール :syuukatuinoue@gmail.com

GitHub :<https://github.com/lovecap7>

Youtube :<https://youtube.com/@loveknotgame>

スキル

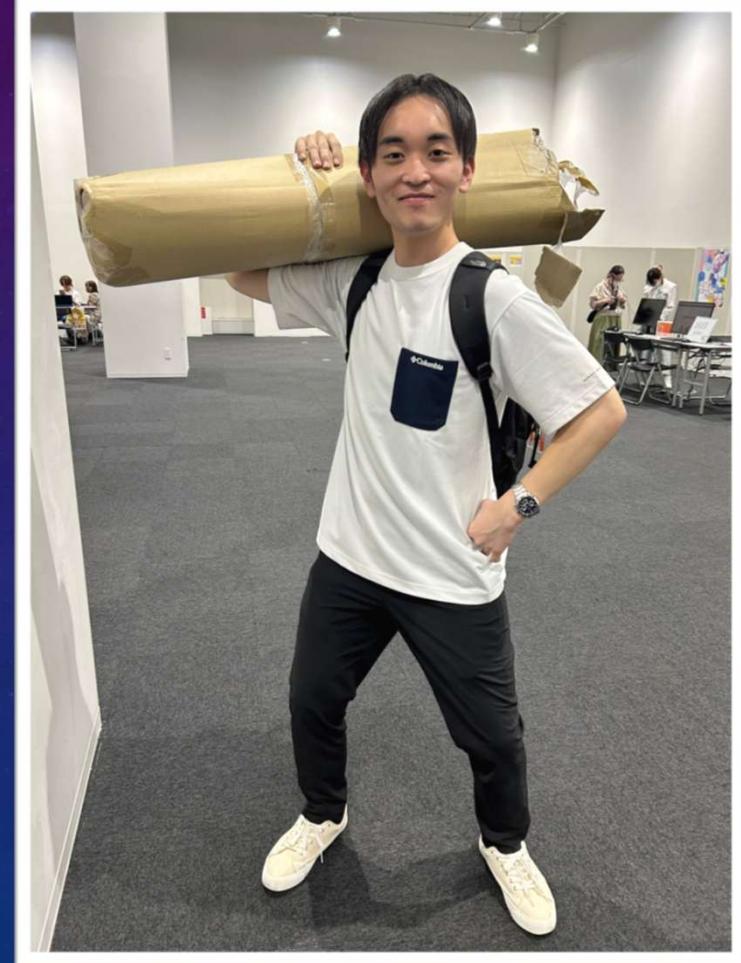
◆ C 3年 ◆ PremirePro 1年

◆ C++ 2年 ◆ Word/Excel 4年

◆ Unity・C# 2年 ◆ PowerPoint 3年

◆ Photoshop 2年 ◆ GitHub 2年

◆ Illustrator 1年 ◆ HLSL 半年



自己PR

チームを支えるリーダーとして今まで頑張ってきました！！

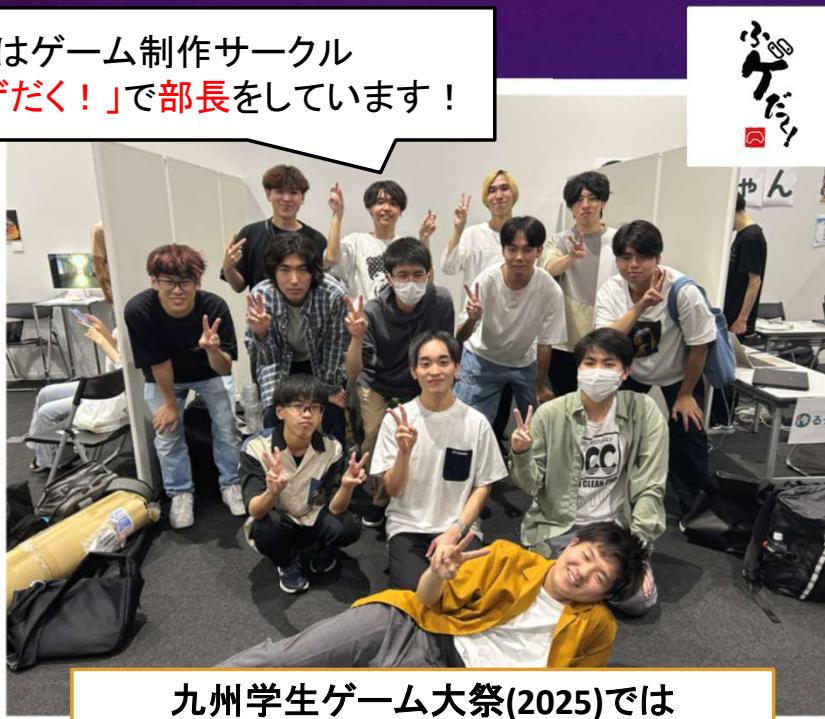
中学生の時はサッカーチームで
ゴールキーパーと副部長をしていました！



応援団もしてました！



現在はゲーム制作サークル
「ふげだく！」で部長をしています！



九州学生ゲーム大祭(2025)では
会場アンケート得票数2位でした！！！



高校生の時はeスポーツ部で
サポート(SUP)と部長をしていました！



高校生限定eスポーツ大会
「Stage:0(2022年)」の
LOL部門で全国出場！！

就活作品



最終作品

作品名 : Project:NeaR

ジャンル : 3Dアクション

開発環境 : DxLib/Visual Studio 2022

言語 : C++

制作期間 : 3ヶ月

制作時期 : 2025年10月～2026年1月

制作人数 : 1人

担当箇所 : プログラム全般

動作環境 : Windows

GitHub : <https://github.com/lovecap7/ProjectNeaR.git>

解説動画 : https://youtu.be/Tkbg_T1ylnM?si=uM6i1smVTQMvqxz3



▼GitHubQRコード



▼解説動画QRコード



ゲーム概要

■ 目標はステージの最後にいるボスを倒すこと

■ 剣と大剣、射撃を駆使して敵を倒そう！！

■ 敵の攻撃をジャスト回避して、そこから
ジャスト回避攻撃を繰り出し大ダメージを
狙おう！！



▼敵を倒しボス部屋を目指す！！



▼いざボス戦！！



▼ボスと激闘！！



▼ボス撃破！！



▼ステージクリア！！



Project:NeaR



ここが面白い①

誰でも簡単にかっこいいアクションができる！！

- ボタン連打でかっこいい連續攻撃ができる
- ジャスト回避が簡単に出せる

▼ボタン連打でコンボ



ジャスト回避攻撃！！▶

◀ ジャスト回避から…



ここが面白い②

個性あふれるボスとの戦闘！！

- ボス毎に攻撃方法や行動パターンが違う
- ボス毎の攻略法を考えるのが楽しい



弾幕を出してくるボス



巨大ボス



第二形態に変身する
ボス



かっこいいアクションで
ボスを撃破せよ！！



技術紹介



技術的に1番頑張ったこと



プレイヤーの手触りを良くすること

そのためにやった技術的挑戦

①先行入力



▼ボタン連打！！



▼入力を保持

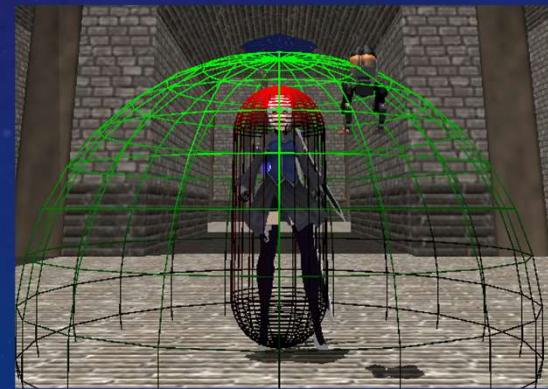


Input

②入力キャンセル



③回避用判定





技術的に1番頑張ったこと:①先行入力

入力を一定フレーム数保持する先行入力の仕組みを実装しました。

行動不可中に行われた入力を保存し、次の行動に遷移可能になったタイミングで処理することで、入力の取りこぼしを防いでいます。

その結果、コンボのつながりが良くなり、快適な操作感を実現しました。

▼ボタン連打！！



▼入力を保持



Input

▼攻撃可能になったら即座に攻撃



▼新しく入力が入った、または入力が残っているか

```
void Input::UpdateBuffer()
{
    //押された入力のフレームを減らしていく
    for (auto& [key, frame] : m_inputBufferFrame)
    {
        //フレームを減らしていく
        if (frame > 0.0f)
        {
            --frame;
        }
    }

    //新しく押された入力をバッファに登録
    for (const auto& keyInfo : m_currentInput)
    {
        if (IsTrigger(keyInfo.first))
        {
            m_inputBufferFrame[keyInfo.first] = kBufferFrame;
        }
    }
}
```

▼攻撃キーの入力が残っていたら攻撃する

```
//攻撃
if (input.IsBuffered("X"))
{
    ChangeState(std::make_shared<PlayerStateLightAttack>(m_player));
    return;
}
if (input.IsBuffered("Y"))
{
    ChangeState(std::make_shared<PlayerStateHeavyAttack>(m_player));
    return;
}
```

技術的に1番頑張ったこと:②入力キャンセル

攻撃・回避・待機などの行動をStateとして管理し、
攻撃後の後隙をキャンセルして次の行動へ
遷移できる仕組みを実装しました。

各行動を状態として分離することで、入力に応じた
遷移制御が明確になり、**テンポの良いアクション**
を実現できました。

▼攻撃後



▼連続攻撃



▼キャンセルフレーム中に入力に応じて遷移

```
//キャンセルフレーム
if ((model->GetTotalAnimFrame() - m_attackData->GetCancelFrame()) < m_frame)
{
    //攻撃の条件
    bool isChargeAttack = m_chargeCountFrame >= kChargeFrame;
    bool isCombAttack = input.IsBuffered("X");
    //武器を持つ
    owner->HaveLightSword();
    //攻撃をするか
    if (isChargeAttack || isCombAttack)
    {
        NextAttack(isChargeAttack, owner, model);
        return;
    }
    //ジャンプ
    if (owner->IsJumpable() && input.IsBuffered("A"))
    {
        ChangeState(std::make_shared<PlayerStateJump>(m_pOwner, m_isWait));
        return;
    }
    //大剣攻撃
    if (input.IsBuffered("Y"))
    {
        ChangeState(std::make_shared<PlayerStateHeavyAttack>(m_pOwner, m_isWait));
        return;
    }
}
```

▼回避



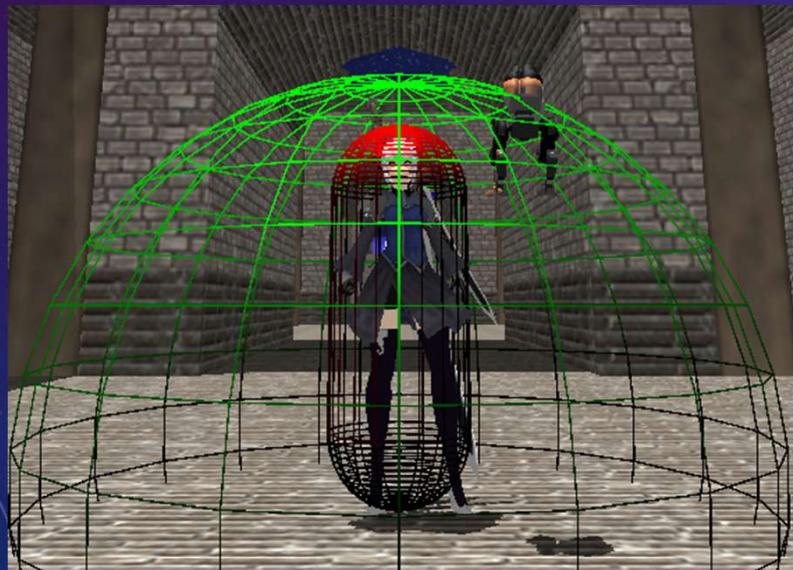
技術的に1番頑張ったこと:③回避用判定

喰らい判定とは別に**回避専用の判定**を実装しました。

これにより、入力タイミングに対する許容範囲が広がり、**ジャスト回避を狙いやすく**なっています。

その結果、プレイヤーがテンポよく行動でき、**よりスタイリッシュなアクション体験**を実現しました。

▼緑の円が回避用判定、赤のカプセルは喰らい判定



これを回避した

▼敵の弾を回避したとしてジャスト回避成功



技術的に2番目に頑張ったこと



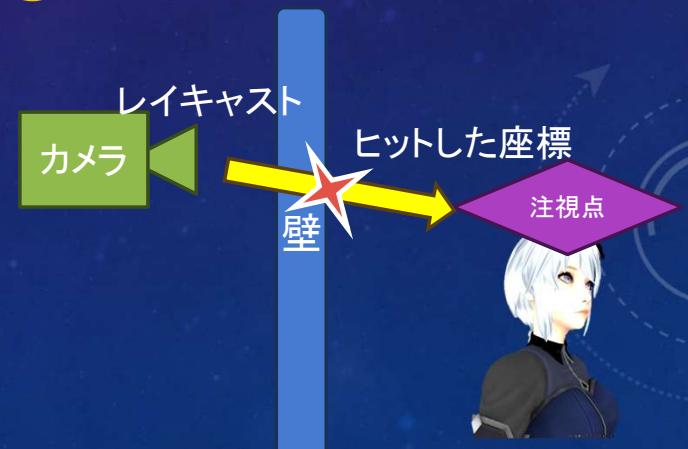
カメラ制御

そのためにやった技術的挑戦

①スタック構造

②ロックオンの拳動

③壁と床との当たり判定



技術的に2番目に頑張ったこと:①スタック構造

CameraController にカメラを **スタック構造** で管理し、常に最後にプッシュされたカメラのみを更新する仕組みを実装しました。

カメラ切り替え時には、**前のカメラの座標や向きを引き継ぐ**ことで、**違和感のないスムーズな遷移を実現**しています。

CameraController

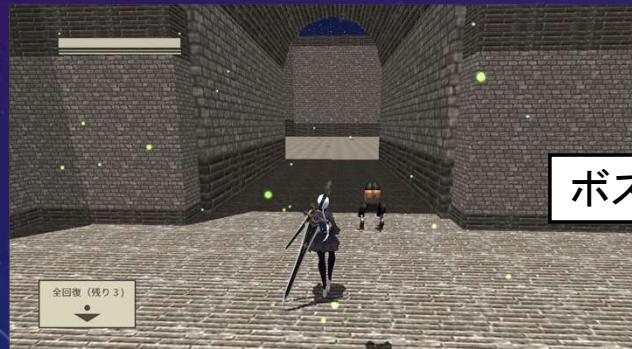
演出カメラ

通常カメラ

更新

(例) 通常のカメラから演出用カメラへ切り替え、演出終了後に元のカメラへ戻る

▼通常カメラ



ボス戦

▼演出カメラ



演出終了

▼通常カメラ



技術的に2番目に頑張ったこと:②ロックオンの挙動(1/2)

ロックオン時には敵の大きさに応じてカメラの高さと距離を動的に調整することで、敵のサイズに依存せず視認性の高い快適な戦闘を実現しました。

▼高さと距離を調整している部分

```
//高いほうに合わせる
basePos.y = MathSub::Max(MathSub::Max(playerPos.y,lockPos.y) , playerPos.y + kUpOffset);
//高さの差で距離を離す(でかい奴ほど引いた視点で見たいので)
basePos -= (toEnemyXZ * (kBackOffset + abs(lockPos.y - playerPos.y) * kBackRate));
//少し横から見たような視点にしたい
basePos += (playerRight * m_lockOnSide);
```

▼通常サイズの敵



▼大きいサイズの敵



技術的に2番目に頑張ったこと:②ロックオンの挙動(2/2)

ロックオン切り替え時には、プレイヤーからターゲットへの向きを基準に右スティックを倒した方向の敵を探索し、その中からカメラの向きに最も近い敵を選択することで、直感的なロックオン切り替えを可能にしています。

(例) 右スティックを左に倒したとき



技術的に2番目に頑張ったこと:③壁や床との当たり判定

カメラの座標から注視点に向けてレイキャストを行い、
その間に壁や床のポリゴンが存在した場合は、ヒットした座標までカメラを
移動させることで、カメラが壁にめり込まないようにしました。



技術的に3番目に頑張ったこと



見た目

そのためにやった技術的挑戦

①シェーダー



②アニメーションブレンド

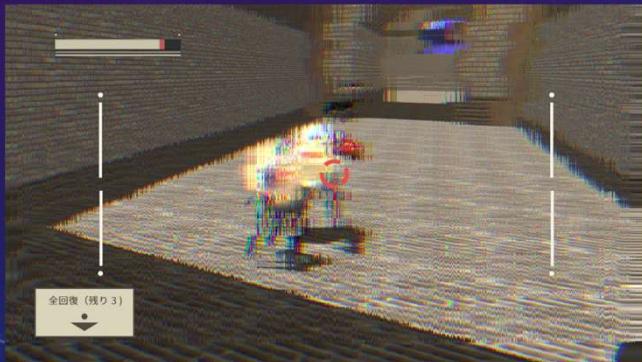


技術的に3番目に頑張ったこと:①シェーダー



HLSL を用いてポストエフェクトを実装しました。
グリッジやモノクロといった機械的な映像表現を加えることで、
無機質でアンドロイドらしい世界観を演出しています。

▼グリッジ



▼ビネット



▼モノクロ



技術的に3番目に頑張ったこと:②アニメーションブレンド



キャラクターの状態に応じた**アニメーションブレンド**を実装しました。
移動・攻撃・回避などの遷移時に補間処理を行うことで、
アニメーションの急な切り替わりを防ぎ、**操作感と見た目の両立**を図っています。

▼前のアニメーション



だんだん変化させる...



▼次のアニメーション





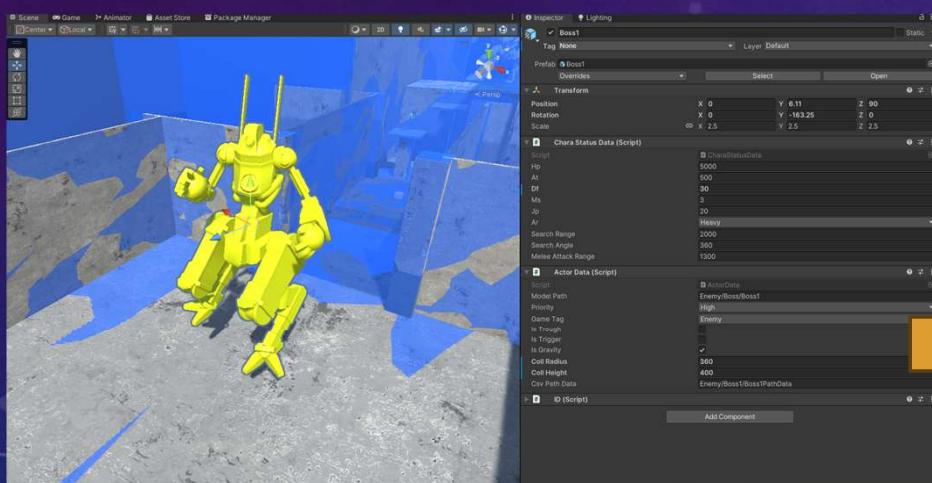
その他の技術的なこと



その他:① Unityをエディターとして使った

Unityで配置したオブジェクトの座標データなどをCSVに書き出して
それを読み込むことでステージやキャラの配置を簡単にできるようになりました。

▼CSV書き出し/読み込み



その他:②グループ化

グループ内で攻撃権を順番に回すことで
プレイヤーを複数の敵が同時に攻撃しないようにしました。

▼画像では各グループをおなかの色で、攻撃権を頭の赤色の球で表しています。



▼グループ内で順番に攻撃権を回す





その他:③外部ファイル化

キャラクターの座標・大きさ・向きや攻撃、アニメーション情報を
外部ファイルで管理しました。

これによってパラメータ調整や敵キャラの量産が効率よく行えるようになりました。

▼アニメーション

状態	武器	アニメーション
Idle	None	RobotArmature Idle
Walk	None	RobotArmature Walk
Run	None	RobotArmature Run
Hit	None	RobotArmature HitReceive_1
HitAir	None	RobotArmature HitAir
Fall	None	RobotArmature Fall
FallHitAir	None	RobotArmature HitAirFall
PunchAttac	None	RobotArmature Punch
JumpAttac	None	RobotArmature JumpAttack
JumpAttac	None	RobotArmature JumpAttackComb
KickRight	None	RobotArmature KickRight
KickLeft	None	RobotArmature KickLeft
ShootAttac	None	RobotArmature Shoot
Down	None	RobotArmature Down
Death	None	RobotArmature Death

▼攻撃

攻撃の名前	多段ヒット	攻撃タイプ	威力	攻撃の重さ	アーマー	ノックバック	上方向の力	再生フレーム	持続フレーム	半径	長さ	前進速度
Punch	FALSE	1	10	3	3	10	0	40	20	200	200	0
KickRight	FALSE	5	10	3	3	10	0	40	10	500	0	0
KickLeft	FALSE	5	10	3	3	10	0	40	10	500	0	0
Shoot	TRUE	2	10	3	3	10	0	50	100	100	0	0
Shoot_2	TRUE	2	10	3	3	10	0	60	100	100	0	0
Shoot_3	TRUE	2	10	3	3	10	0	70	100	100	0	0
Shoot_4	FALSE	2	10	3	3	10	0	80	100	100	0	0
JumpAttac	FALSE	5	10	3	3	6	0	60	10	800	0	2
JumpAttac	TRUE	5	10	3	3	6	0	80	10	800	0	0
JumpAttac	TRUE	5	10	3	3	6	0	120	10	800	0	0
JumpAttac	FALSE	5	10	3	3	6	0	150	10	800	0	0

▼各ステージ情報

名前	ID	アクターID	座標X	座標Y	座標Z	回転X	回転Y	回転Z	大きさX	大きさY	大きさZ
Player	147	Character	0	0	0	0	0	0	0	1	1
Boss3	290	Character	0	10	130	0	180	0	4	4	4
Robot1	273	Character	0	2.73	31.81	0	180	0	0.3	0.3	0.3
Robot1	36	Character	0	2.22	55	0	180	0	0.3	0.3	0.3
Robot1 (1)	338	Character	5.15	5.97	62.48	0	180	0	0.3	0.3	0.3
Robot1 (2)	138	Character	-5.35	5.83	60.74	0	180	0	0.3	0.3	0.3
Robot2	288	Character	2	4	66.02	0	196.75	0	0.3	0.3	0.3
Robot2 (1)	161	Character	-2	4	65.96	0	196.75	0	0.3	0.3	0.3

作品総括

まだまだ戦闘の要素が少ない

上記の問題を解決するためには

- ・プレイヤーの攻撃手段を増やす
- ・プレイヤーのアクションに合わせた敵
- ・ステージギミックに合わせた戦闘



制作実績 個人製作①

作品名 : BRAVE BLADE

ジャンル : 3Dアクション

開発環境 : DxLib/Visual Studio 2022

言語 : C++

制作時期 : 2025年6月～2025年9月

頑張ったことと学んだこと

- ・ クラス設計の大切さ
- ・ 敵の挙動



作品名 : ColorfulFighter

ジャンル : 2D対戦格闘

開発環境 : DxLib/Visual Studio 2022

言語 : C++

制作時期 : 2024年11月～2025年2月

頑張ったことと学んだこと

- ・ ステートパターンを使ったプレイヤーの状態遷移
- ・ 対戦ゲームのバランス調整の難しさ

制作実績 個人製作②

作品名 : ハンバーガーランナー

ジャンル : 3Dランアクション

開発環境 : Unity

言語 : C#

制作時期 : 2024年7月～2024年9月

頑張ったことと学んだこと

- ・ ゲームを作る楽しさ
- ・ 人にプレイしてもらうことの大しさ



作品名 : 物理系スナイパー 井上ラビット

ジャンル : 3Dアクション

開発環境 : DxLib/Visual Studio 2022

言語 : C++

制作時期 : 2025年3月～2025年4月

頑張ったことと学んだこと

- ・ 3Dモデルを使った制作
- ・ 3Dカメラの挙動・制御

制作実績 チーム製作①

作品名 : SAMURAI BUSTERS

ジャンル : 3D協力アクション

開発環境 : Unity 言語 : C#

制作時期 : 2025年4月～2025年5月

制作人数 : プログラマー 3人、プランナー 1人

担当箇所 : 敵AI全般、クリアやタイマー等のゲームルールシステム

頑張ったことと学んだこと

- ・ 短期間での制作をする上での計画・スケジュールの組み立て
- ・ 敵AIの作り方・考え方



作品名 : ダーウィン★ダービー

ジャンル : 2Dアクション

開発環境 : Unity 言語 : C#

制作時期 : 2025年10月4日～2025年10月5日

制作人数 : プログラマー 4人、プランナー 2人、デザイナー 2人

担当箇所 : プレイヤー

頑張ったことと学んだこと

- ・ ゲームジャムでの制作経験
- ・ 他の職種(担当)の人と制作をする際に要望を伝えたりすることが難しかった



制作実績 チーム製作②

作品名 : 超次元戦車

ジャンル : 3D対戦アクション

開発環境 : Unity 言語 : C#

制作時期 : 2024年9月～2024年10月

制作人数 : プログラマー 3人

担当箇所 : プレイヤー、アイテム、勝敗等のゲームシステム、UI

頑張ったことと学んだこと

- ・ マルチプレイヤー対応
- ・ 戦車の独特的な操作感



作品名 : エッグいDoragon

ジャンル : 2Dシューティングアクション

開発環境 : Unity 言語 : C#

制作時期 : 2024年5月～2024年7月

制作人数 : プログラマー 5人、デザイナー 1人

担当箇所 : 2体の子供ドラゴン選択シーン

頑張ったことと学んだこと

- ・ 初めてのチーム制作の経験
- ・ 全体の進捗を把握することの大変さ



今後の目標

キャラクター挙動を担うプログラマーを目指す。

今回の制作でプレイヤーの手触りを良くしていくことがゲームの遊びやすさと面白さに繋がることに気づきました。

これからは物理や数学的知識を学習しつつ、プレイヤーが直感的に操作できるキャラクター表現を追求していきたいです。