

シン モヒト クマル
SINGH MOHIT KUMAR

ゲームプログラマー



京都コンピュータ学院駅前校
デジタルゲーム学系 ゲーム開発基礎科
卒業年度： 2027年見込み
電話番号： 070-8997-0383



自己紹介

出身

インド出身です。2024年4月に来日し、ゲーム開発の学習と制作取り組んでいます。

自己PR

私は独学でUnityを学び、来日後から本格的にゲーム制作に取り組んできました。理解した技術をすぐ実装に活かすことが得意で、個人制作だけでなく、学校内チームや GGX Game Jam などでもプログラマーリーダーを担当しています。

制作では、メンバー全員が成長できるようタスクを調整し、必要に応じてコード改善や技術アドバイスも行っています。判断力と問題解決力を活かしながら、より良いゲーム体験を届けるために日々成長しています。

自己紹介

ゲーム開発で大事にしていること

- ・読みやすく保守しやすいコードを書くこと
- ・パフォーマンスに強い実装を心がけること
- ・ゲームとしての遊びやすさを常に意識すること
- ・デザイナーやプランナーが調整しやすい 柔軟な設計 を作ること

現在取り組んでいること

- ・Unreal Engine の習得 (Blueprint と C++ を用いたゲーム制作)
- ・DirectX を使ったゲーム開発の基礎理解
- ・3Dモデリングの学習 (自作ゲーム用のキャラクターモデルや環境モデル制作)
- ・シェーダーの理解強化
- ・ゲーム制作に必要な数学・物理の復習



スキル

メインスキル

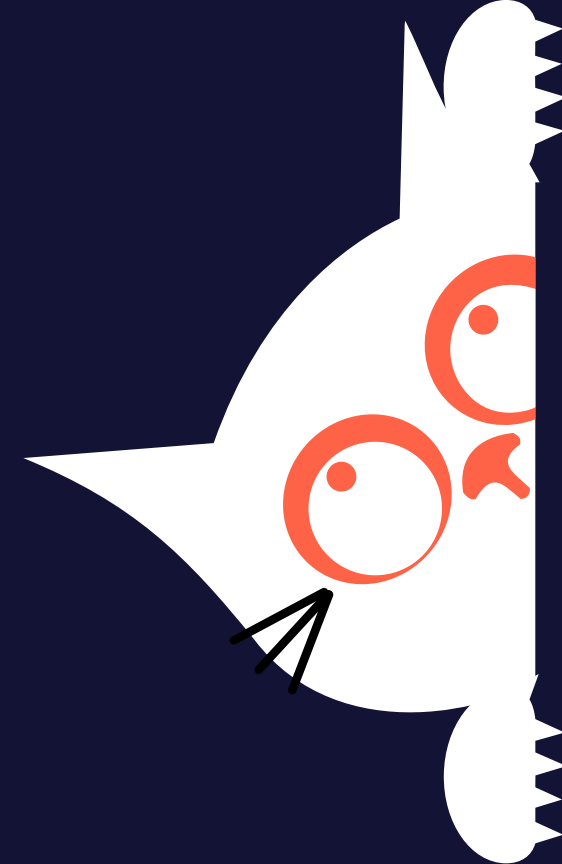
- C# 
- C++ 

- Unity 
- Unreal Engine 
- 3Dモデリング 

補足スキル

- JavaScript / TypeScript
- Next.js
- React
- Node.js
- MongoDB
- MySQL

- UI/UX（基礎）
- Express.js
- AWS（基礎）
- GitHub / Git

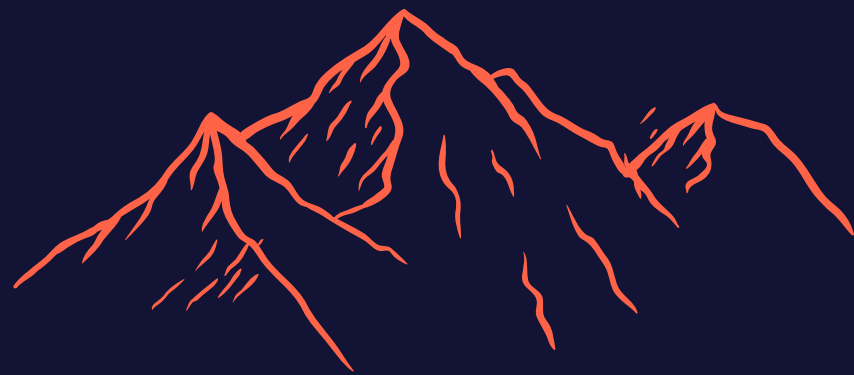


好きなこと・影響源



趣味

・ハイキング



自然の中を歩きながら気分をリフレッシュするのが好きで、景色を見ながらゆっくり散策する時間を大切にしています。

・ドライブ



知らない場所を走るのが好きで、特に長距離ドライブで新しい景色や場所に触れることが楽しいです。

好きなこと・影響源

ゲーム

- ・よく遊ぶジャンル
アクション、シューター、レース、ストーリー重視の作品、オープンワールド
- ・好き・影響を受けたタイトル
 - ・GTA シリーズ (特に GTA SA / GTA 5)
 - ・GTA SA：昔の作品なのにマップが非常に広く、ジムなどの膨大な遊び要素・隠れた mechanics が魅力的
 - ・GTA 5：Euphoria 物理エンジンによるリアリズムが刺激になり、自分のゲーム制作でも「GTA SA の膨大なディテール × GTA5 のリアリズム」を目標にしている
 - ・The Last of Us 1 & 2 / Uncharted
 - ・ストーリー性の強いゲームへの憧れから、来日以前から構想していた「インドの有名な神話（ラーマヤナ）を題材にしたゲーム」を作りたいという気持ちを強く後押しした
 - ・SEKIRO / Ghost of Tsushima / Tekken など好き



好きなこと・影響源



アニメ

- ・よく見るジャンル

ダークファンタジー、心理系、アクション

- ・好き・影響を受けた作品

アカメが斬る！ / 86 / 屍鬼 / 東京喰種 / 黒の契約者 /
青春ブタ野郎シリーズ / どろろ / Another / 学校の怪談 / ひぐらしのなく頃に /
僕だけがいない街 / ベルセルク / プランダラ / MONSTER / 少女戦記 / Fate シリーズ /
ダーリン・イン・ザ・フランキス / 空の境界 / ノラガミ /
絶園のテンペスト / 境界の彼方 / ブギーポップは笑わない / 終わりのセラフ など多数



好きなこと・影響源



映画

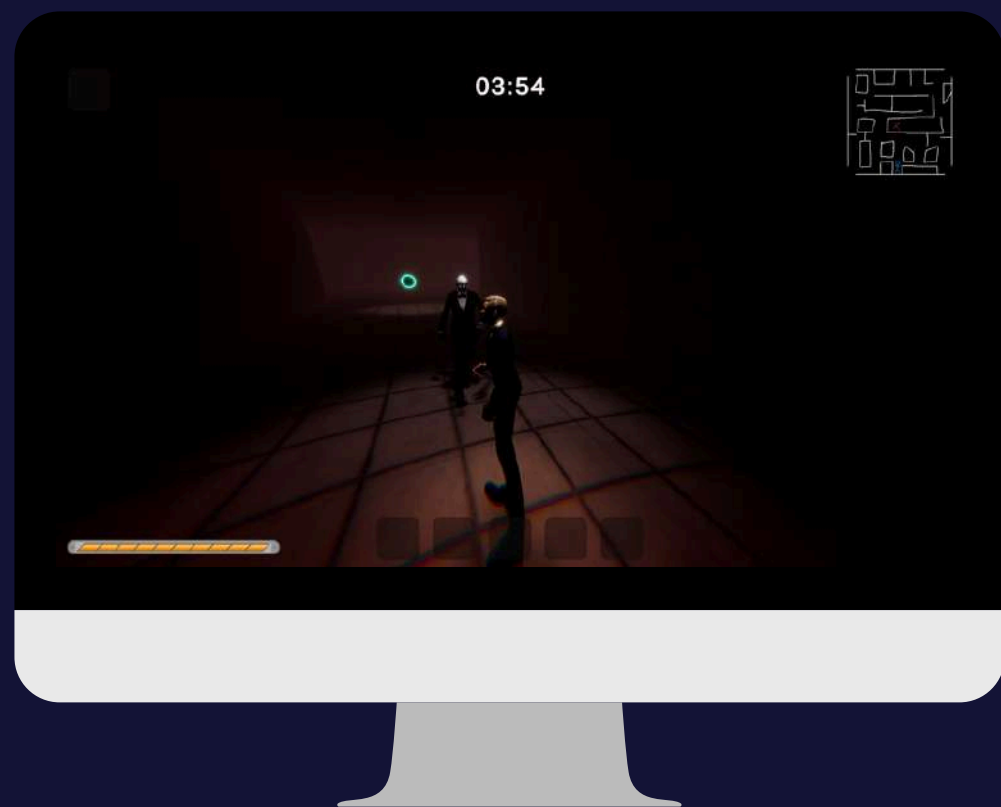
- ・よく見るジャンル
アクション、ダーク系心理映画
- ・好き・影響を受けた作品
 - ・ Blade (三部作)
→ 子どもの頃から何度も見返している代表的な作品
 - ・ ゾンビ / アポカリプス映画全般
→ コロナ期間中はほぼ存在する全てのゾンビ映画を観るほど大好き



外部活動・イベント参加

BitSummit ボランティアスタッフ (2025)

国内最大級のインディーゲームイベントにて、ボランティアスタッフとして会場サポート業務を担当しました。
開発者や来場者の方々と関わる中で、ゲーム業界の運営体制や現場の雰囲気について学ぶことができました。



11月祭 ゲーム制作・展示 (2025)

サークル活動としてオリジナルゲームを制作し、学校の展示会に出展しました。
開発を担当し、来場者からのご意見や反応を通して、ユーザー視点でのゲーム改善の重要性を実感しました。

GGX Game Jam (2025)

現在、プログラマーリーダーとしてタスク管理や技術面の調整を行いながら、チーム開発を進めています。

サークル活動 (Dory)

役割：プログラマー

私が所属しているゲーム制作サークル「Dory」では、定期的な勉強会やサークル内ゲームジャム、他サークルとの合同制作など、幅広い活動を行っています。チーム開発ではGit運用・コードレビュー・技術共有を意識しながら、メンバーと協力して作品づくりに取り組んでおります。

また、日本に来てから「友達を作るのが難しい」と感じていましたが、サークル活動を通して自然とコミュニケーション力が向上し、仲間も増えました。チーム制作は技術面だけでなく、人とのつながりや協力の大切さを実感できる貴重な経験となっています。



作品詳細①

U-Turn



作品詳細①

U-Turn

開発期間 : 2日

ジャンル : カーレース

ゲーム説明:

加速と減速を使い分けながら周回コースを走り抜けるレースゲームです。スピードの出しすぎはカーブでコースアウトし、ゲームオーバーになります。制限時間内に規定周回を完了するとクリアです。

操作方法 : スペースキー : 加速 (離すと減速)

Unity

自作



作品詳細①

苦勞した点

スピードとカーブ角度からコースアウト判定を行う挙動作りに最も苦勞しました。

技術解説

- Unity Spline を使用してコースを構築
- Rigidbody による自前の車挙動（SplineAnimate は未使用）
- Spline の進行方向ベクトルからカーブ角度を算出し、危険速度と比較してコースアウト判定
- Spline Mesh Collider をコースに適用し、物理走行が可能
- 速度の加減速制御（Space キー：加速／離すと減速）
- カーブの危険度に応じたゲームオーバー処理
- メインライトのクッキーに、カスタムレンダータクスチャ（シェーダー）で作成した疑似雲を使用

```
1 reference | Windsurf: Refactor | Explain | Generate Documentation | X
float GetTurnAngle(float tPos)
{
    float checkDist = 1f; // Distance to check ahead/behind on the spline
    float tAhead = Mathf.Clamp01(tPos + checkDist / splineLength); // a bit ahead on the spline
    float tBehind = Mathf.Clamp01(tPos - checkDist / splineLength); // a bit behind on the spline

    // World positions on the spline
    Vector3 aheadPos = spline.EvaluatePosition(tAhead);
    Vector3 behindPos = spline.EvaluatePosition(tBehind);

    // Direction vectors relative to the car's current position
    Vector3 forwardAhead = (aheadPos - transform.position).normalized;
    Vector3 forwardBehind = (transform.position - behindPos).normalized;

    // Angle between these directions represents the turn's sharpness
    return Vector3.Angle(forwardBehind, forwardAhead);
}
```

```
// Check steepness for possible failure
if (CanGoOffTrack())
{
    SendVehicleOffTrack();

    /*
     * ⇒ Decide off track velocity for different angles
     * 0° → perfectly straight section.
     * 10°-20° → gentle curve.
     * 30°+ → sharp turn.
     * 60°+ → U-turn territory.
     */
}
```

作品詳細①

処理フロー図

ゲームフロー

Start

初期化

Update

入力処理

スペース押下 → 加速

スペース解除 → 減速

速度更新

スプライン移動

カーブ角度チェック

危険速度？

YES → コースアウト → ゲームオーバー

NO → 続行

ラップチェック

ラップ完了？

YES → 次のラップ / 最終ならクリア

NO → 続行

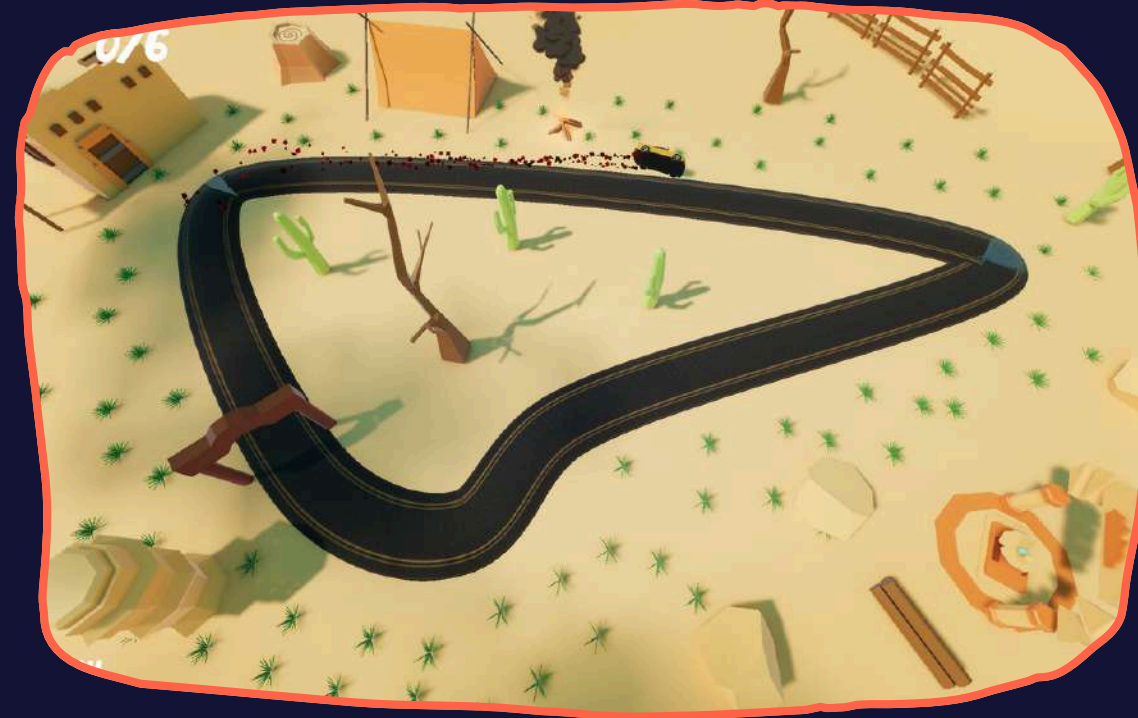
タイマー確認

タイムアップ？

YES → ゲームオーバー

NO → 続行

ループ



作品詳細②

Ghost Protocol - Escape



作品詳細②

Unity

自作



Ghost Protocol - Escape

開発期間 : 5日

ジャンル : 2Dアクション / シューティング

ゲーム説明:

本作は、正体が露見したスパイエージェントが敵地からの脱出を目指す、2Dアクションゲームです。四方八方から出現する敵軍のヘリコプターの攻撃を回避・迎撃しながら、制限時間内に脱出地点へ到達することが目的となります。

操作方法 :

- W/A/S/D 移動
- マウス移動 照準
- 左クリック 射撃
- Shift 急ダッシュ
- Space ジャンプ
- Space + 壁 壁ジャンプ



作品詳細②

苦勞した点

初挑戦のピクセルアート制作と、射撃方向に応じて上半身を回転させるプレイヤーリギングの実装。



技術解説

- ・ピクセルアートを自作し、アニメーション用にパーツ分けして制作
- ・プレイヤーキャラクターをボーン構造でリギングし、上半身のみを回転可能に設計
- ・マウスの照準方向を取得し、射撃角度に応じて上半身をリアルタイムで回転
- ・壁との接触判定と入力タイミングを利用した壁ジャンプ処理を実装
- ・ダッシュ・ジャンプ・射撃を組み合わせた高機動アクション設計
- ・画面外から敵が出現することで、常にプレッシャーを与えるゲーム進行を構築



作品詳細②

処理フロー図

ゲーム開始

- プレイヤー初期化
 - 位置・体力のリセット
 - 武器・照準の初期化
 - カメラ・UI設定

敵ヘリコプター生成処理

- 画面外からスポーン
- 一定間隔で追加生成

ゲームループ

- プレイヤー入力処理
 - 移動・ダッシュ
 - ジャンプ・壁ジャンプ
 - 照準・射撃
- プレイヤー制御更新
 - 上半身回転・アニメーション更新
- 敵AI処理
 - プレイヤー追跡
 - 射撃処理
 - 突撃処理
- UI・制限時間更新

勝敗判定

- 脱出地点到達 → クリア
- プレイヤー死亡 → ゲームオーバー

ゲーム終了

操作方法

↑/↓/←/→	移動
マウス移動	照準
左クリック	射撃
Shift	急ダッシュ
Space	ジャンプ
Space + 壁	壁ジャンプ

向かってジャンプし、接触した瞬間に再ジャンプすると壁ジャンプできる。
タイミングよく移動キーを使えば、より高く登れる。



ENTER

空から追撃が始まる。

RESTART

QUIT

闇に消えて、
存在しなかったことになった



作品詳細③

Killzone Zero



作品詳細③

Unity

自作



Killzone Zero

開発期間 : 2週間

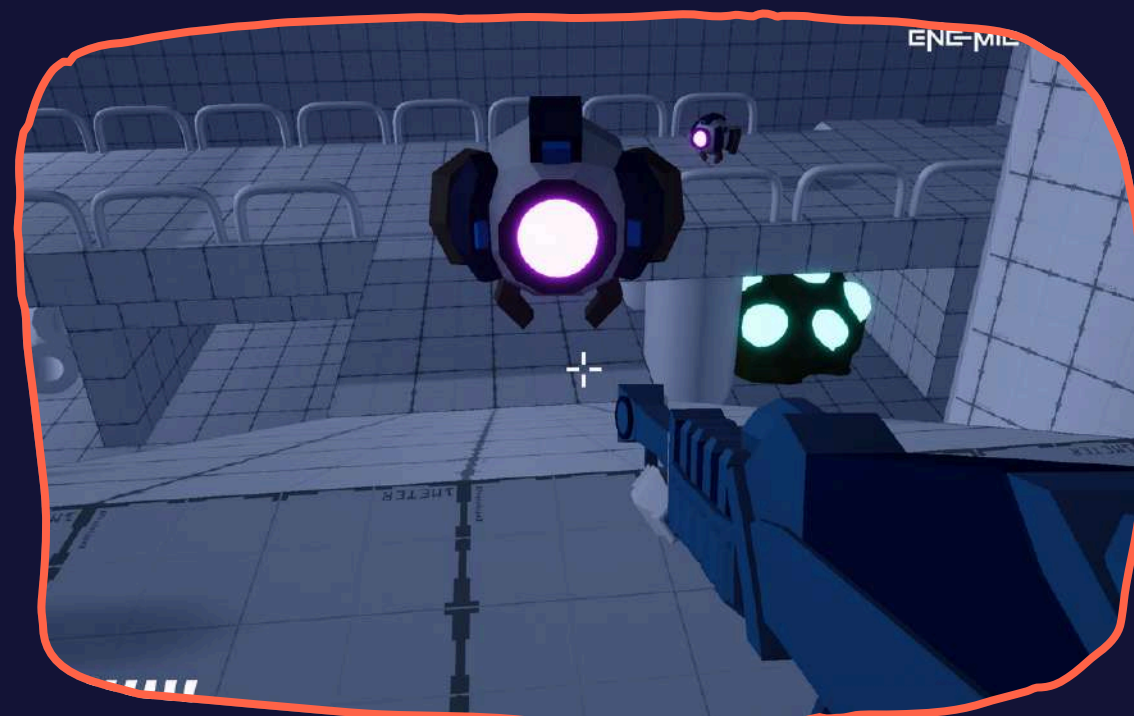
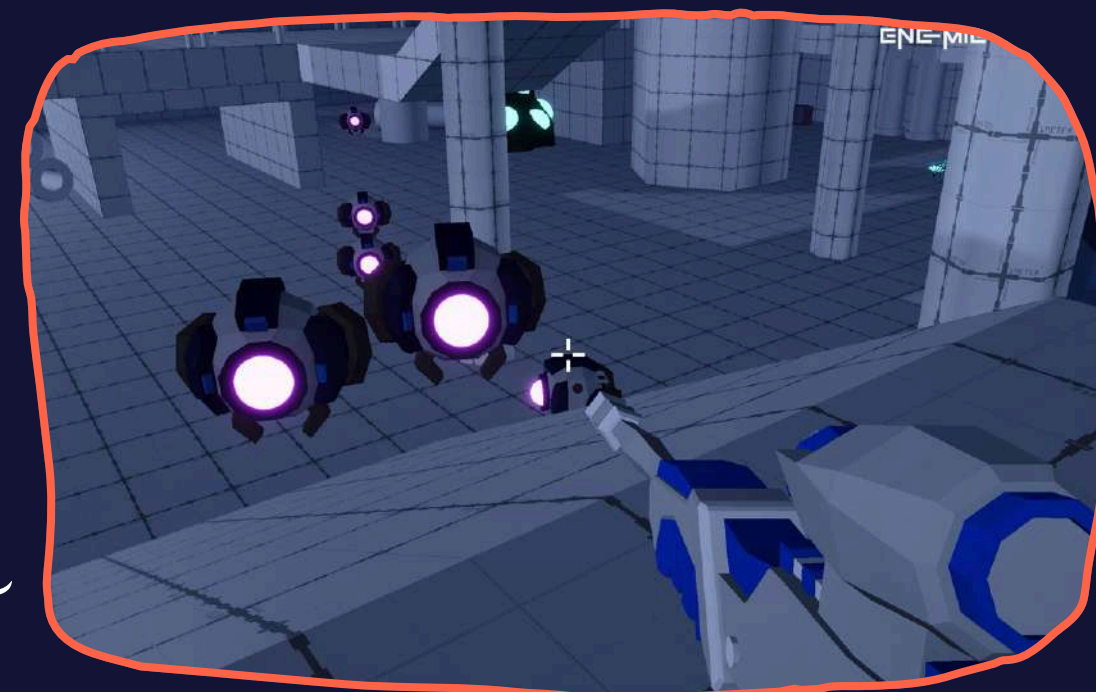
ジャンル : FPS / ウェーブ生存シューティング

ゲーム説明:

次々と迫るロボットのウェーブを撃破し、生き残りを目指すFPSゲーム。
マップ内の武器や弾薬を確保し、限られたライフで最後のウェーブまで耐え抜こう。

操作方法 :

- W/A/S/D 移動
- Space ジャンプ
- マウス移動 視点操作
- 左クリック 射撃
- Ctrl (長押し) スナイプースコープ使用 (スナイパーのみ)



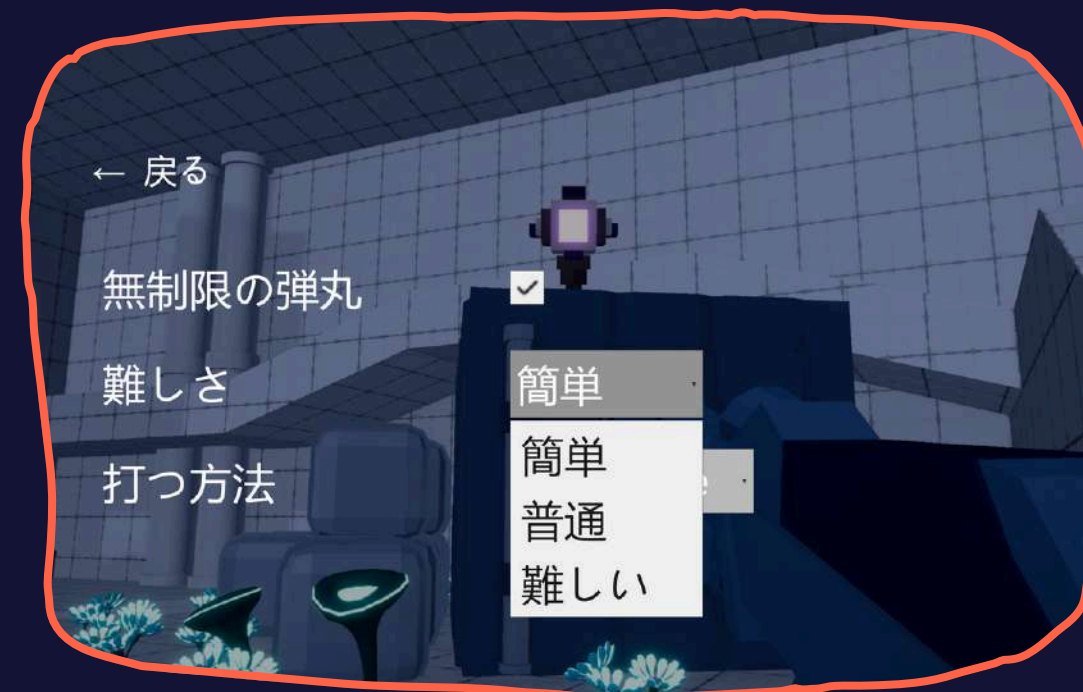
作品詳細③

苦勞した点

難易度や射撃方式を自由に切り替えられる柔軟なシステムを作ること。

技術解説

- ・ 難易度設定が変更可能
- ・ 射撃方式の切替対応（Raycast / Projectile をプレイヤー設定からスイッチ可能）
- ・ 柔軟なゲームバランス管理（ScriptableObject を用いたパラメータ管理）
- ・ 拡張性を意識したアーキテクチャ（武器システム・敵 AI をモジュール化）
- ・ 難易度ごとに敵ウェーブ数・敵数・出現間隔を自動生成し、ゲームテンポを動的に調整
- ・ プレイヤーは HP 0 でも残ライフがある限りランダム安全地点へ自動リスポーン
- ・ 普通難易度以上でのみ作動する自動攻撃タレットを配置



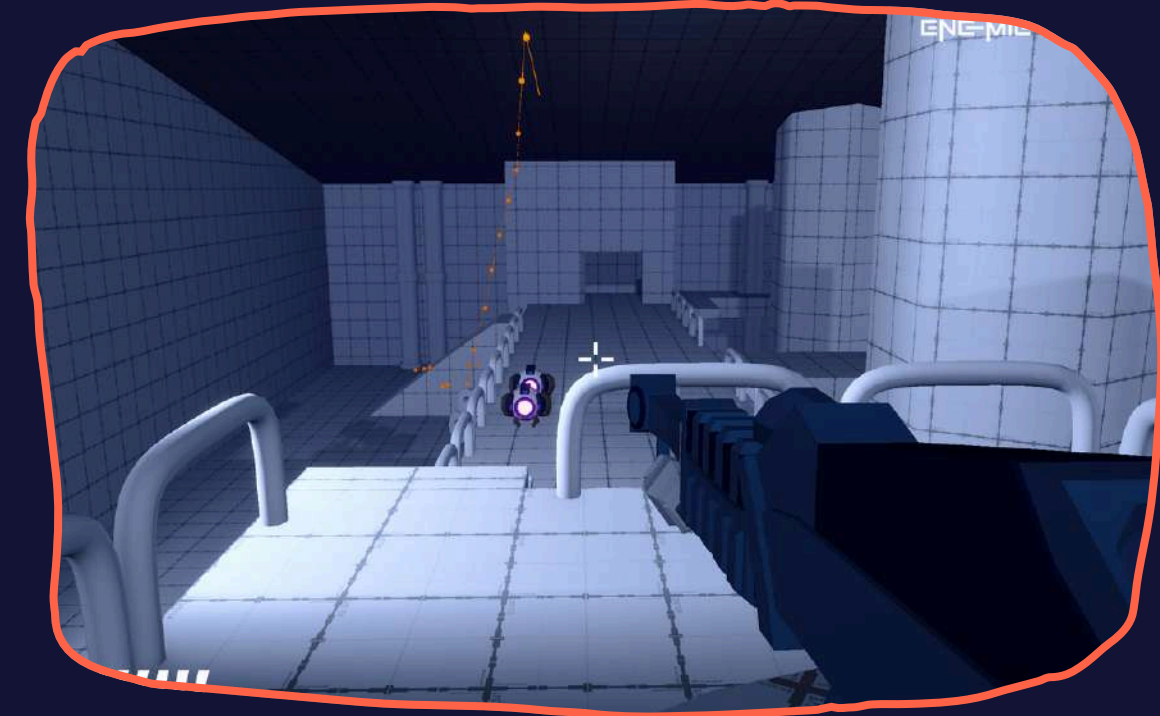
作品詳細③

処理フロー図

ゲーム開始

- ▶ オプション設定読込
 - └ 弾薬モード設定 (通常 / 無限弾)
 - └ 難易度設定 (Easy/Normal/Hard)
 - └ 射撃方式設定 (Raycast / Projectile)
- ▶ 敵ウェーブ自動生成 (難易度に応じて変化)
- ▶ プレイ開始
 - └ ▶ プレイヤー入力処理
 - └ ▶ 攻撃処理 (弾薬消費 or 無限弾)
 - └ ▶ 敵スポーン & AI処理
 - └ ▶ タレット (Hard以上のみ)
 - └ ▶ HPチェック
 - └ HP > 0 → 続行
 - └ HP = 0 → ライフ減少
 - └ ライフ残り → ランダム安全地点へリスポーン
 - └ ライフ無し → ゲームオーバー
 - └ ▶ ウェーブクリア判定
 - └ 次ウェーブへ
 - └ 全ウェーブ終了 → リザルト

ゲーム終了



作品詳細④

Brainless Zombies



作品詳細④

Unity

チーム



Brainless Zombies

開発期間 : 2日

開発人数 : 4名

ジャンル : 1人称視点 × カラー判定アクション

自分の担当 : プログラマーリーダー

レベルデザイン、ゾンビ移動処理、脳みそ判定・挙動、ゲームロジック全般などUI 以外のほぼ全システムを担当。

ゲーム説明 :

暗いステージで迫りくるゾンビに、同じ色の脳みそを素早くぶつけて人間に戻すアクションゲーム。照明が点いたり消えたりする中、視点を切り替えながら色を判別し、より多くのゾンビを救うことを目指す。間違った色を当てると一瞬ハマるが抜ける。ゾンビに近づかれすぎるとゲームオーバー。ゾンビを救うほど速度が加速し、難易度が徐々に上昇する。

操作方法 :

- Space 視点切り替え
- ← → キー 脳みそを弾く (左右)



作品詳細④

苦勞した点

前後から同時に迫るゾンビと脳みそを、どちらの方向へ撃つべきか瞬時に判別させるシステム設計に苦戦しました。

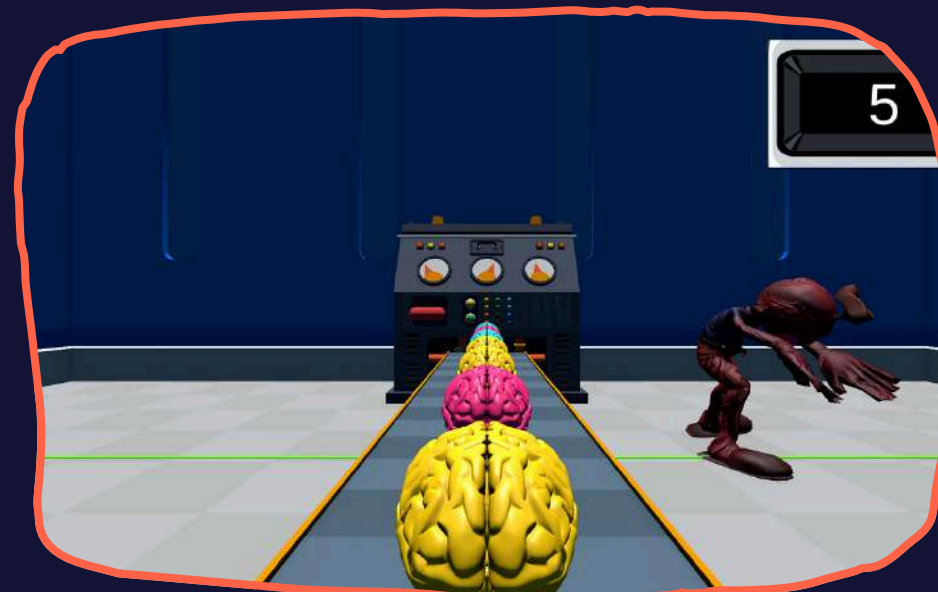
```
_brains.Remove(brain);

references | Windsurf: Refactor | Explain | Generate Documentation | X
private void ShootBrain(ZPosition zPlayerPosition, XPosition xPosition)
{
    if (zPosition == zPlayerPosition && _brains.Count > 0)
    {
        ZombieInZone zombieInZone = _shootableZoneManager.FindZombieToShoot(
            zPlayerPosition,
            xPosition,
            _brains[0].BrainColor
        );

        // shoot at the zombie
        if (zombieInZone.Transform != null && zombieInZone.rb != null)
        {
            BrainController brain = _brains[0];
            brain.ShootItSelf(zombieInZone.Transform.position + new Vector3(0f, 2.8f, 0f));
            _brains.Remove(brain);
        }
    }
}
```

技術解説

- ・ 前方・後方のゾンビを独立して管理するため、2キュー構造の判定システムを実装
- ・ 脳みそとゾンビの色判定（C/Y/M）を enum 化し、処理を分岐せずシンプルに実装
- ・ 問題解決①：どちら側のゾンビに撃つべきか分かりにくい
→ 前後それぞれに「最も近いゾンビ」を常に取得する優先度システムを実装
- ・ 問題解決②：押したキーと反対側のゾンビへ撃ってしまう
→ 現在の視点方向と入力方向を同期し、意図した側へ確実に脳みそが飛ぶ仕様へ統一
- ・ 照明の点滅を0.3~1秒ランダムで実装し、視認性と緊張感を演出
- ・ ゾンビ救済数に応じて 1.2倍ごとに速度上昇する加速システムを実装
- ・ Unity 6 の新InputSystemでキーボード+コントローラー両対応に設定



作品詳細④

処理フロー図

ゲーム開始

Start

初期化（速度・色・方向・UI）

ゾンビ／脳みそ生成開始

Update

入力処理

視点切り替え（前／後）

脳みそ発射（左右）

色判定

ゾンビ色 == 脳みそ色 → 変身成功

不一致 → 影響なし

移動処理

ゾンビ前進

脳みそコンベア移動

距離判定（前／後）

一番近いゾンビを優先ターゲットへ設定

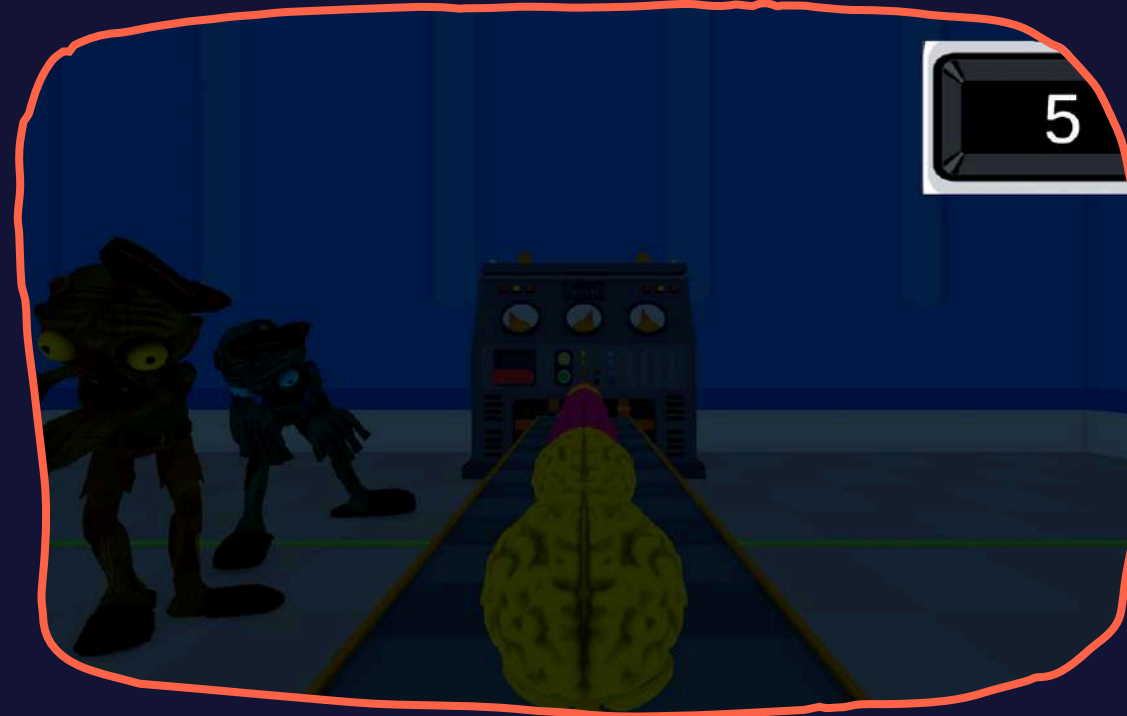
加速処理

5人救済ごとに速度1.2倍

ゲームオーバー判定

ゾンビが一定距離内に到達

結果画面



作品詳細⑤

歪ノ夢路



作品詳細⑤

歪ノ夢路

開発期間 : 3~4 週間

開発人数 : 5名

ジャンル : ホラーアクション / 探索

自分の担当: プログラマーリーダー

プログラム全体を中心担当として、以下を主に実装・管理しました。

- ・プレイヤー関連の全機能
(移動、当たり判定、ダメージ処理、体力管理、リスポーンシステムなど)
- ・敵 AI (巡回・発見・追跡など)
- ・ゲームループ構築 (開始 → プレイ → ゲームオーバー → リトライ)
- ・ポストプロセッシング効果 (ホラー演出、ダメージ演出など)
- ・パフォーマンス改善 (不要処理削減、最適化)
- ・ギミックシステム実装
- ・アイテムシステム
- ・プログラマー管理 (タスク配分、進捗管理)
- ・コードレビュー・修正指導
- ・様々な教育 (Github など)



Unity

チーム



作品詳細⑤

ゲーム説明：

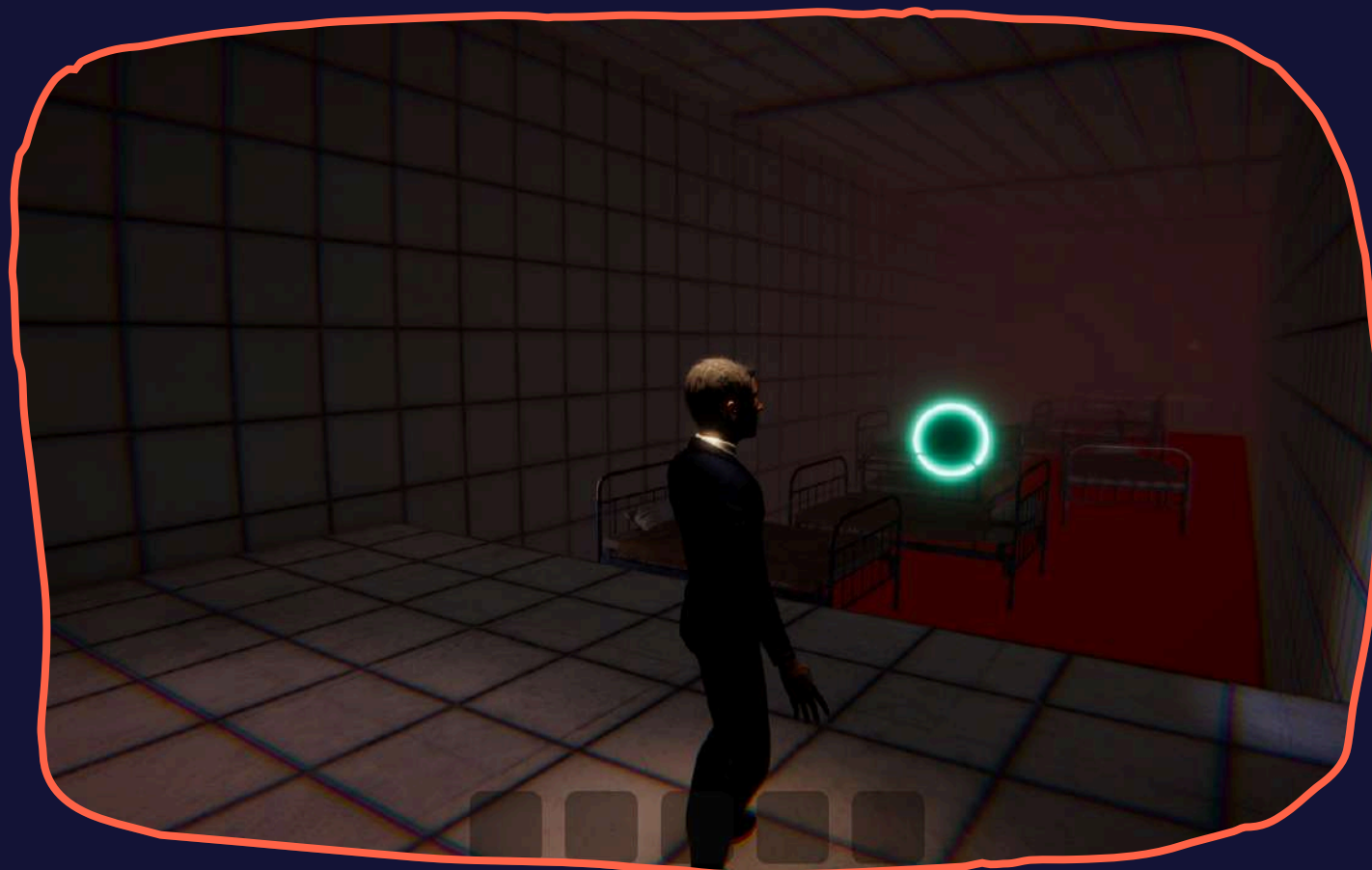
プレイヤーは悪夢のように歪んだ空間を進み、幽鬼めいた怪異から逃げながら出口を目指します。怪異は視覚・距離を元にプレイヤーを探索し、発見すると高速で追跡してきます。

画面の歪み・ノイズ・暗所演出などによる恐怖感と、限られた視界の中でルートを探し続ける緊張感がメイン要素です。

操作方法：

- ・ W/A/S/D 移動
- ・ Space ジャンプ
- ・ マウス移動 視点操作
- ・ Shift 走る（スプリント）

※コントローラー使用可能が、未テスト



作品詳細⑤

苦勞した点

動く床の上でプレイヤーを正しく立たせる物理挙動の安定化に苦勞しました。

技術解説

- ・動く床でプレイヤーが滑る問題の修正：Rigidbody ベースの移動処理で、床の毎フレームの移動量を算出し、その差分をプレイヤーへ加算する方式に変更。これにより、動く床に乗っても滑らず安定して立てるように改善
- ・プレイヤーのリスポーンシステム：死亡 → リセット を自動化するフローを構築し、演出とゲームループの継続性を両立
- ・ポストプロセス最適化：Bloom・Vignette などを使用
- ・汎用ギミックシステムの構築：スイッチ・トリガー・移動床などをprefab化し、レベルデザイナーが配置するだけで挙動が動く仕様にして開発効率を改善
- ・コードレビューと GitHub の導入サポート：チームメンバーに GitHub の Branch 運用・PR・コードレビューのやり方をレクチャーし、開発フローを安定化

```
// 動く方向
direction = axis == ObjectPropogatorAxis.X ? Vector3.right :
}

0 references | Windsurf: Refactor | Explain | Generate Documentation | ✕
void Update()
{
    // オブジェクトを動かす
    MoveObject();

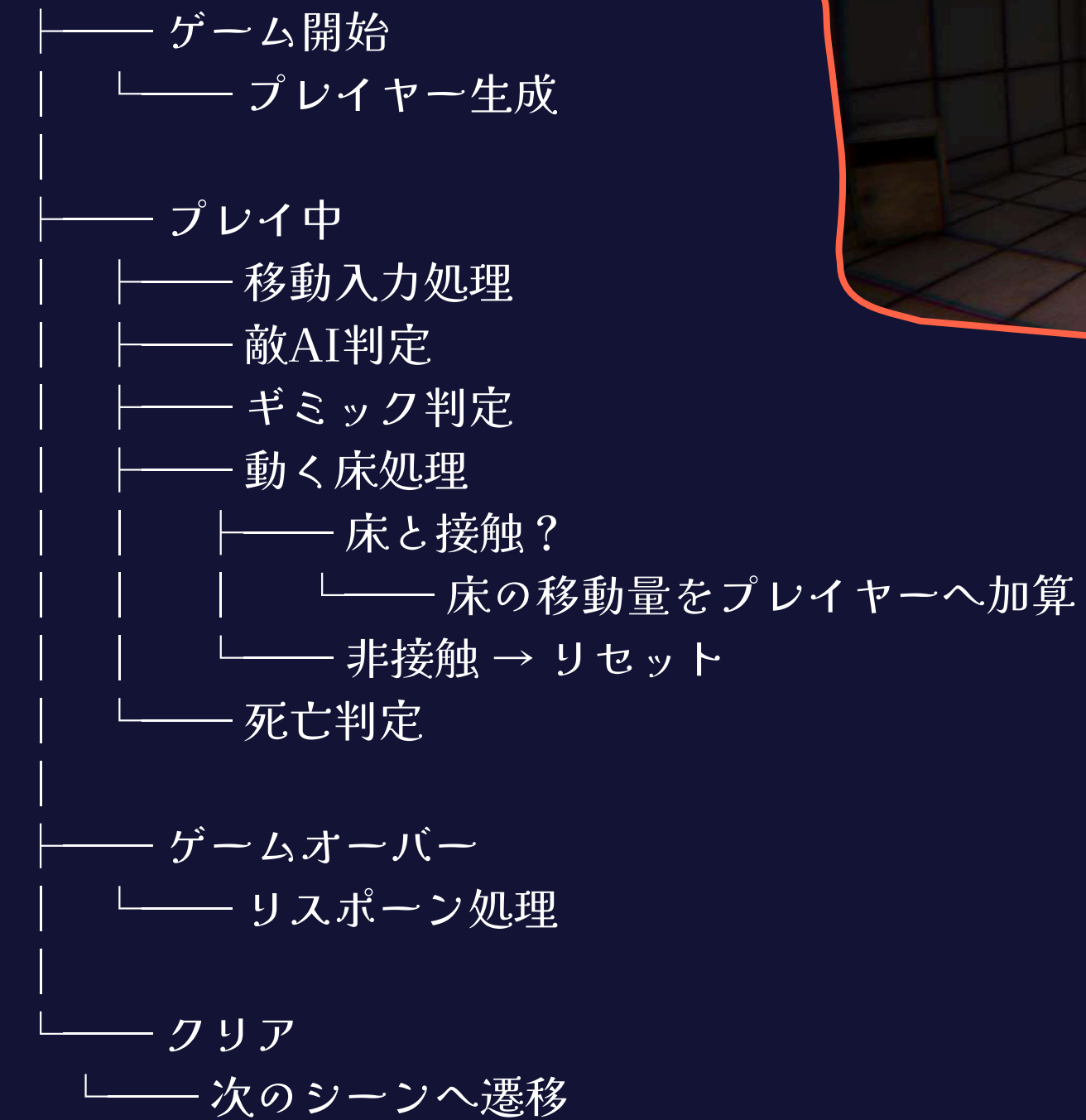
    // ** プレイヤーがプラットフォームと一緒に動くようにするためのもの **
    // 速度を求める
    platformVelocity = (transform.position - lastPos) / Time.deltaTime;
    platformVelocity.y = 0; // Y軸の速度は無視する
    lastPos = transform.position;
}
```



作品詳細⑤

処理フロー図

ゲームフロー



作品詳細⑥

Horizon Survivor



作品詳細⑥

Unity

自作



Horizon Survivor

開発期間 : 2週間

ジャンル : 3D サバイバルアクション

ゲーム説明 :

プレイヤーは“サッカーボール”となり、街中を徘徊する人々や動物から逃げながら、夕暮れまで生き残ることを目指すサバイバルゲームです。時間経過で減少する耐久値や、キック回数の残量を管理しつつ、マップ上の酸素缶やツールキットを利用して生存時間を伸ばします。

操作方法 :

- W/A/S/D 移動
- Space ジャンプ
- マウス移動 視点操作
- Shift 走る (スプリント)
- Alt ブースト (ツールキットが持っている場合)



作品詳細⑥

苦勞した点

大量の徘徊NPCと膨大なウェイポイントにより、同時生成でゲームが一時停止してしまう問題に直面しました。

技術解説

- ・ 敵のスポン処理を バッチ方式 に変更し、一度に大量生成せず数体ずつ時間差で生成してフリーズを防止
- ・ 敵ごとのウェイポイント数を 最大5個までに制限 し、無駄な経路計算を抑えて処理負荷を大幅削減
- ・ 全NPCの視野判定を角度・距離・レイキャストの3段階チェックに分離
- ・ オプションから難易度を変更でき、難易度が上がるほど NPC の視野拡大・検知時間短縮・追跡速度上昇で発見されやすくなる
- ・ さらに 昼の長さが延び、生存時間が増加し、耐久回数の減少と NPC 数増加でプレッシャーが大幅に強化
- ・ 全体的に、難易度に応じて探索・回避の緊張感が段階的に高まるゲームバランスを構築

```
/* 敵をバッチで生成する  
* 各バッチの間に少し待つ  
* パフォーマンスを向上させる  
*/  
1 reference | Windsurf: Refactor | Explain | ×  
IEnumerator SpawnInBatch()  
{  
    int no_of_batches = 10;  
    int enemies_per_batch = totalEnemies / no_of_batches;  
  
    for (int i = 0; i < no_of_batches; i++)  
    {  
        SpawnEnemies(enemies_per_batch);  
        yield return new WaitForSeconds(5f);  
    }  
  
    yield return null;  
}
```



作品詳細⑥

処理フロー図

ゲーム開始

- プレイヤー初期化
 - 位置・体力・耐久値リセット
 - カメラ/UI 設定
- NPCスポン処理
 - バッチ単位で敵生成
 - 各敵にウェイポイント割当（最大5）
- ゲームループ
 - プレイヤー操作入力
 - NPC巡回・追跡判断
 - キック判定・演出
 - 体力/耐久値更新
 - 夕暮れ進行
- 勝敗判定
 - 体力または耐久値が0 → 敗北
 - 夜になるまで生存 → 勝利

ゲーム終了



作品詳細⑦

聖地からの解放

開発中 Unity チーム



作品詳細⑦

聖地からの解放

開発期間 : 3~4 週間

開発人数 : 8名

テーマ : 聖地

ジャンル : 2D アクション／弹幕アクション

※GGX GAME JAM

自分の担当：プログラマーリーダー

プログラム全体を中心担当として、以下を主に実装・管理しました。

- ・プレイヤーの移動処理
- ・ゲーム全体のゲームループ設計
- ・言葉弹幕を用いたバトルシステムの実装
- ・ポストプロセス演出の実装
- ・プログラマー間のタスク分担・進行管理

開発中

Unity

チーム



作品詳細⑦

ゲーム説明：

本作は、「言葉」を通じて幽霊と向き合う 2D アクションゲームです。
プレイヤーは“聖地”のような空間を舞台に、幽霊が放つ言葉の弾幕を回避・選択しながら進んでいきます。

幽霊の攻撃は「寂しい」「お腹が空いた」など感情を表す言葉で構成されており、プレイヤーは適切な言葉を返すことで、幽霊を攻撃するのではなく“理解し、落ち着かせる”ことを目的としています。

操作方法：

- ・ W/A/S/D …………… 移動
- ・ Space …………… 言葉への反応
- ・ マウス移動 …………… 選択肢決定



作品詳細⑦

苦勞した点

技術的な難易度以上に、オンライン環境でのチーム管理が最も大きな課題でした。



今後の予定

- ・ゴーストを倒すのではなく、条件を満たすことで仲間として加入するシステムの実装
- ・仲間になったゴーストが、次のステージでプレイヤーを支援する仕組みの追加
- ・展示会向けに、初見プレイヤーでも理解しやすいUI・演出・チュートリアル改善



作品詳細⑧

Unveil

開発中 Unity 自作



作品詳細⑧

Unveil

開発期間 : 約1ヶ月

ジャンル : 一人称視点探索ホラー (FPS / 3D)

ゲーム説明:

本作は、通常では見えない「ドア」や「敵」が存在する遺跡を探索するゲームです。

プレイヤーはペイントボールを投げることで、当たった場所や幽霊の輪郭を可視化することができます。

しかし、ペイントが当たった幽霊はプレイヤーを認識し、追跡・攻撃を開始します。

プレイヤーは逃げる・戦うといった行動を使い分けながら、10分という制限時間内に宝物のある「見えない扉」を探し出す必要があります。



開発中

Unity

自作



作品詳細⑧

操作方法：

- ・ W/A/S/D 移動
- ・ Space ジャンプ
- ・ Shift ジャンプ
- ・ マウス移動 視点操作
- ・ 左クリック ペイントボールを投げる
- ・ Rキー 銀斧を使った攻撃

攻撃システム：

プレイヤーは Rキーの入力回数 によって異なる攻撃を行うことができます。
Rキーを 1回・2回・3回 連続で押すことで、それぞれ異なる攻撃が発動します。

- ・ 1回押し：通常攻撃
- ・ 2回押し：強攻撃
- ・ 3回押し：特殊攻撃

入力回数の判定には一定の受付時間を設けており、
素早い入力操作によって状況に応じた攻撃を選択できる設計になっています。



作品詳細⑧

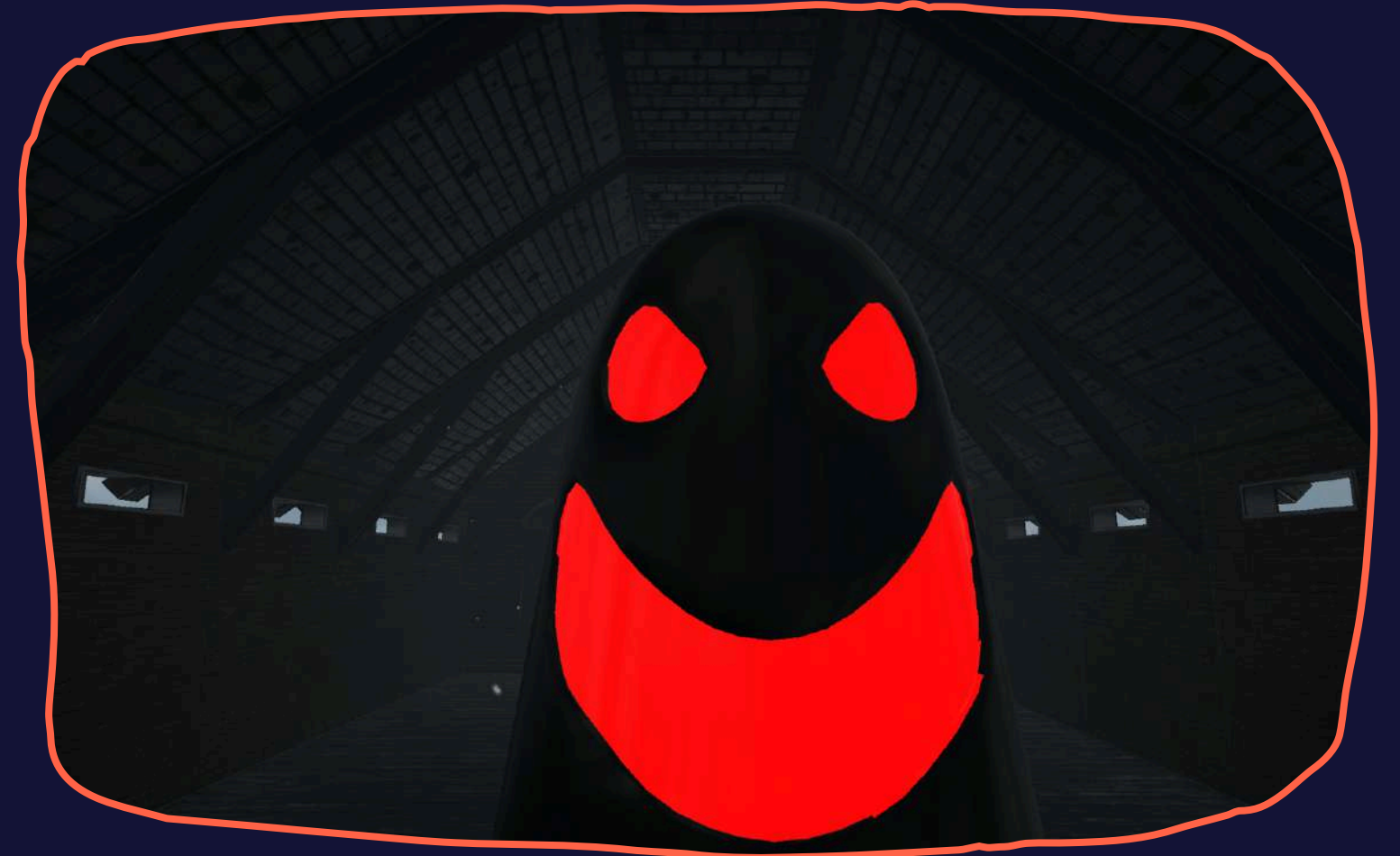
苦勞した点

透明オブジェクトに対して、不可視状態を維持したままペイントによる可視化表現を実現する点に最も苦勞しました。



今後の予定

- ・プレイヤーの状態を視覚的に伝えるUIの実装
- ・ゲーム開始・クリア時のカットシーン演出の追加
- ・ゲームテンポや難易度バランスの調整



作品詳細⑨

Deliver or Defend

開発中

Unity

自作

作品詳細⑨

Deliver or Defend

開発中

Unity

自作

開発期間 : 約2ヶ月～

ジャンル : 3D ラグドール物理ベースアクションゲーム

ゲーム説明:

本作は、夜の街を舞台にした配達アクションゲームです。
プレイヤーは人間ではなく、宇宙から来た配達員として、暗い街の中で荷物を届ける仕事をしています。

配達中、街のあちこちから盗賊が現れ、プレイヤーの持つ荷物を奪おうとします。プレイヤーは盗賊を避けながら進むこともできますが、危険な場面では荷物を投げて撃退することも可能です。ただし、荷物はそのままスコアや達成条件に直結するため、守るか、使うかの判断が重要になります。
キャラクターはラグドール構造で表現されており、物理挙動による不安定さや偶発的な動きも含めて、

「思い通りにいかないが、操作していて面白い」体験を目指しています。



※Steam公開を想定しているため、モデルやエフェクトなどはすべて自作しています。

今後の更新について

現在、他の制作物も順次追加準備中です。
今後も更新を続けてまいります。



ご覧いただき、ありがとうございました。
引き続き技術向上と作品制作に励んでまいります。

