

# スカイレガリア -空島の鍵と厄災の謎-

～Game Development Summary～

2025. 12. 12

日下 拓海

## 目次

1. 基本情報
2. ゲーム概要
3. 技術概要
4. 開発プロセス
5. まとめ

## 1. 基本情報

### ゲーム基本情報

ゲームタイトル：スカイレガリア -空島の鍵と厄災の謎-

コアコンセプト：空中に浮かぶ足場をジャンプと飛行で渡り歩き、ステージ上の鍵を集め、回避を駆使してボスモンスターに挑む 3D アクション

対応プラットフォーム：PC (Windows/macOS)

使用ツール：Unreal Engine (v5.4.4)

制作期間：2025. 7 - 2025. 12

所要時間数の概数：200～250 時間程度

動作環境：Windows 10 / 11、macOS

制作人数：1 人

ゲーム開発における役割：全工程を担当しました

## ゲームアピールポイント（3点）

- ・ 任意方向に移動できる「飛行」をコアアクションとして採用しています。一定時間飛行できるため、敵に追われている時に空中に逃げたり、高低差のある足場や通常のジャンプでは届かない場所を探索したりと、3D空間を活かした行動の選択肢が広がります。

- ・ 敵の攻撃をジャスト回避すると「チャンスタイム」が発生し、数秒間無敵になり、敵の速度が低下します。その間は一方的に攻撃できるため、一気に大ダメージを与えられます。通常時の慎重な立ち回りと、チャンスタイム中のラッシュとのギャップによって、無双感と爽快感のある戦闘体験を狙っています。

- ・ 飛行時に足場のオン／オフが切り替わったり、2Dと3Dの視点が入れ替わったりと、「飛行」と連動したギミックを各ステージに配置しています。これにより、いつ飛ぶかを考えながら進む、立体的なステージ攻略が楽しめます。

## 開発者情報

開発者名： 日下 拓海

所属： 弘前大学理工学部電子情報工学科

連絡先：[takumi.090414528@gmail.com](mailto:takumi.090414528@gmail.com)

## プレイ動画／紹介動画

<https://youtu.be/fu8ed8akTVA?si=C06KpIzIu0hdjZ-P>



本作のプレイ映像と、ストーリー、戦闘システム等に関する紹介をまとめた動画です。

## 2. ゲーム概要

### 基本ルール

まずジャンプや飛行を駆使して、各ステージ上に配置された5つの鍵を集めます。すべての鍵を集めるとボスモンスターが出現し、そのボスを倒すことでクエストクリアとなります。最初のチェックポイントはスタート地点ですが、鍵を取るとチェックポイントが鍵の位置になります。落下した際には一定ダメージを受けてしまうので落ちないように気をつけて探索する必要があります。もし落ちてしまった際には直前のチェックポイントからリスタートします。

戦闘時は移動・ジャンプ・攻撃に加えて、任意方向に移動できる飛行やジャスト回避からのチャンスタイムを活用しながら、敵の攻撃を避けつつダメージを与えていきます。敵からダメージを受けたり、落下ダメージを受けたりしてHPが0になるとゲームオーバーになります。ゲームオーバー時は拠点に戻るか、チェックポイン

トからやり直すかを選択できます。

## コアゲームループ

拠点でクエストを受注→クエストへ出撃→飛行アクションを使ってステージ上の鍵を5本集める→鍵を集め終わると敵が出現→ジャスト回避からのチャンスタイムを活用して敵を撃破→拠点に帰還し、次のクエストが解放される

## 操作方法

### ゲームパッドの場合（開発者推奨）

Lスティック：移動

Rスティック：視点移動

L：飛行

R：ダッシュ

A：アタック（長押しでチャージアタック）／インタラクション

Y：ステータス上昇魔法

X：ジャンプ

B：回避（長押しでチャージ回避）

ZR：魔法切り替え

### キーボードの場合

W/A/S/D：移動

トラックパッド：視点移動

Q：飛行

E：ダッシュ

F：アタック（長押しでチャージアタック）／インタラクション

C：ステータス上昇魔法

SPACE：ジャンプ

R：回避（長押しでチャージ回避）

V：魔法切り替え

## 実際のゲーム画像



飛行状態



チャンスタイム



ステージに散らばる鍵

## プレイ時間の目安(プレイヤーの習熟により異なります)

想定クリア時間：2時間程度

## 想定プレイヤー像

- ・3Dアクションでキャラクターを動かしながら、敵を倒したりステージを攻略するのが好きな方
- ・空中移動や「飛行」を使って、高低差のあるステージを探索したり、ギミックを解いていく遊びが好きな方
- ・短い時間で1クエストずつ区切って遊べるゲームが好きな方

※詳しいルールや実際のプレイは、紹介動画で解説しています。

### 3. 技術概要

#### 使用技術

エンジン/フレームワーク：

- ・ Unreal Engine (v. 5.4.4)

スクリプト：

- ・ Blueprint

敵の行動：

Behavior Tree + AIController で実装

開発環境：

- ・ クロスプラットフォーム対応 (Windows / macOS) / Unreal Editor (v5.4.4)

ビルド・ツール類：

- ・ Unreal Engine 標準ビルドツール
- ・ Git / GitHub

使用した外部アセット (モデル・背景)：

- ・ プレイヤーキャラクター： Paragon: Aurora
- ・ 敵キャラクター： Paragon (Grux, Narbash, Rampage, Sevarog), ROG\_Creatures
- ・ 背景・ステージ： Medieval Dungeon, Sky Town, Stylized Provencal
- ・ 補足： 上記アセットは外見およびモーションの基礎として使用していますが、キャラクターの移動ロジック、攻撃判定、敵の AI (Behavior Tree)、エフェクトの発生タイミング等はすべて自作スクリプトで制御しています。

#### 主なシステム・モジュール

- ・ PlayerCharacter (BP\_Aurora)：

プレイヤーの移動、ジャンプ、飛行など、操作まわりのロジックを集約したブループリントです。Enhanced Input からの入力を各アクション用イベントに振り分け、地上／飛行状態やチャンスタイム中かどうかといったフラグに応じて移動速度や当たり判定を切り替えています。ダメージ処理では HP の減少に加え、ジャスト回避の成功判定やヒットストップ、チャンスタイム発動もここで制御しています。

- ・ EnemyBase (BP\_Enemy)：

敵キャラクターの親ブループリントで、HP や当たり判定、待機／攻撃アニメーションの再生を共通で管理します。プレイヤーとのオーバーラップやカスタムイベントをトリガーに、ランダムで待機モーション／攻撃モーションを選択しつつ、攻

撃時には当たり判定や効果音のオン／オフを制御します。ダメージ処理では HP が 0 になったタイミングで撃破演出を再生し、GameInstance にクリアフラグを設定してステージクリア処理へとつなげています。

- ・ GameMode (BP\_ThirdPersonGameMode) :

1 クエスト中のゲーム進行を管理するブループリントです。ステージ開始時のプレイヤー生成や UI 初期化、イントロ／チュートリアル再生、NPC との会話からクエスト開始・終了を制御します。鍵（コイン）の取得数が条件を満たしたタイミングでボス出現演出やカットシーンを再生します。

- ・ GameInstance (BP\_GameInstance) :

ゲーム全体を通して保持される進行状況を管理するブループリントです。受注可能なクエスト一覧やクリア済みフラグ、ストーリー進行用の各種フラグを保持し、レベル間で状態を共有します。また SaveGame と連携してセーブ／ロード処理の中核を担い、タイトル画面でロードすることで前回の続きから再開できるようにしています。

- ・ UI / HUD (各種 WBP) :

HP バー、鍵の所持数表示、チュートリアルなど、画面上の UI を管理します。プレイヤーや EnemyBase、GameMode からのイベントを受け取り、ゲージの更新やテキスト表示などを通じてゲーム状況をフィードバックします。

## 設計の方針・意識したこと

- ・ UE 標準クラスを活かした責務分離

Unreal Engine のフレームワークに沿って、プレイヤー操作は PlayerCharacter、1 クエスト中の進行管理は GameMode、ゲーム全体をまたぐ進行状況は GameInstance、永続データは SaveGame といった形で役割を分けました。これにより、「どの処理をどこに書くべきか」が明確になり、ステージ追加やセーブ仕様の変更がしやすい構成を意識しています。

- ・ 親ブループリントによる共通化と拡張性

敵キャラクターは EnemyBase を親ブループリントとして定義し、HP 管理や当たり判定、待機／攻撃アニメーション再生などの共通処理を集約しました。そのうえで個々の敵ごとにパラメータや攻撃パターンを設定するだけでバリエーションを増やせるようにし、仕様変更の影響範囲を最小限に抑えています。NPC も同様に共通

の親 BP を用意し、会話やクエスト開始の処理を再利用できるようにしています。

- ・ Blueprint の可読性とデバッグ性の重視

PlayerCharacter や GameMode などノード数が多くなりがちなブループリントでは、機能ごとに関数／カスタムイベントへ切り出し、コメントと枠（リルートノード・コメントボックス）で「移動」「飛行」「ダメージ処理」「会話」「カットシーン」などのまとまりが一目で分かるように整理しました。また、フラグやタイマー名も機能が推測しやすい名前を付けることで、後から挙動を追いやすくなるよう意識しています。

- ・ 飛行アクションとステージギミックを前提とした設計

本作の特徴である「飛行」を活かすため、移動床やトランポリン、飛行中だけ有効になる床などのステージギミックは個別の Blueprint で実装しつつ、PlayerCharacter の状態フラグ（飛行中かどうか）をトリガーに動作するよう統一しました。これにより、「いつ飛ぶか」「どの高さを通るか」をプレイヤーに考えさせるステージ設計を行いやすくしています。

## 4. 開発プロセス

### 開発プロセスの概要(タイムライン)

2025. 07-09 : 企画・プロトタイプ作成

Unreal Engine 5 の基本的な操作や Blueprint のワークフローを学びながら、「飛行」を中心としたプレイヤーアクションと、敵のビヘイビアツリーによる挙動のプロトタイプを作成しました。この段階では鍵集めなどのアクション要素を入れる予定はなく、スタート時に出現する敵を倒してクエストをクリアしていくシンプルな 3D アクションとして構成していました。

2025. 09-10 : 鍵・ステージ実装

敵の出現条件として鍵集めの要素を取り入れました。それに伴い、空中に足場を配置したステージを実装しました。同時に、GameMode／GameInstance を用いたクエスト進行管理や、HP バー・鍵カウンタなどの UI も整備しました。

2025. 10-11 : ステージギミックとチャンスタイム、演出の追加

移動床やトランポリン、飛行時にオン／オフが切り替わる床などのステージギミックを用いたステージを複数追加し、飛行アクションと連動したステージ構成にブラッシュアップしました。また、敵の攻撃をジャスト回避することでチャンスタイムが発生する仕組みを導入し、その際のスロー演出やエフェクトを調整すること

で、戦闘時のメリハリと手触りを強化しました。

## 2025. 11-12 : デバッグ・バランス調整・ビルド

通しプレイを繰り返しながら、プレイヤーの攻撃力、敵の体力などを調整し、遊びやすさと緊張感のバランスを整えました。あわせて、UI 表示の微修正やバグ修正、パッケージビルドの確認を行い、提出用として一通りプレイ可能な状態まで仕上げました。

## 開発体制・担当範囲

開発体制 : 個人開発

担当範囲 : 企画、ゲームデザイン、設計、実装、テスト、デバッグまで、ゲーム開発の全工程を一人で担当しました。

※GitHub 上では開発環境の都合で、Windows 実機検証用に協力者のアカウントをコラボレーターとして追加していますが、実装・作業内容はすべて本人によるものです。

## ワークフロー・ツール

・バージョン管理 :

Git / GitHub を使用しました。

・タスク管理 :

Notion を使用しました。タスクを優先度ごとに整理し、重要なものから順に実装を進めました。また、開発中に発生した課題とその解決方法を随時まとめるほか、開発スケジュールの管理にも活用しました。

## 開発における課題とその解決

開発中に直面した課題とその解決について、課題→解決→効果→学びという4つの観点から紹介します。

### 課題① : 飛行アクションが強すぎてゲームバランスを崩していたこと

課題 :

当初は飛行に時間制限を設けておらず、常に自由に飛び続けられる仕様でした。

その結果、離れた足場にも簡単に届き、落下リスクもほとんどないため、ステージ攻略が大幅に簡単になってしまっていました。飛行時間だけ制限しても、回避を連発することで依然として遠くまで移動できてしまうという問題が残っていました。

解決：

飛行を「強力だがコストの高い行動」と位置づけ直し、飛行時間に上限を設けると同時に、飛行中は徐々にHPが減少するようにしました。さらに、回避を行うたびに残り飛行時間を消費する仕様に変更し、回避を多用しても到達距離が極端に伸びないように調整しました。

効果：

飛行し続けると時間とHPの両方を消費するため、「常に飛んでいれば安全」という状態から「ここぞという場面で飛行を切る」使い方へと変化しました。プレイヤーは地上移動と飛行を状況に応じて使い分ける必要が生まれ、ステージ構成の意図も伝わりやすくなりました。

学び：

強力なアクションを設計する際は、単に性能を下げるのではなく、リソース消費やリスクとのトレードオフを明確に設計することが重要だと学びました。今後は、新しいアクションを追加する際にも「どれくらいの頻度で使われるべきか」「どんな代償を払うべきか」をセットで考えるようにしていきたいです。

## 課題②：敵が攻め続けてきて、回避の手応えも分かりにくかったこと

課題：

開発初期の敵は常にプレイヤーを追いかけて、ほぼ休みなく攻撃する挙動になっていました。そのため、プレイヤーは回避だけで手一杯になり、反撃する余裕がほとんどない「ひたすら逃げるだけ」の戦闘になっていました。また、回避成功時の演出も地味で、「本当に避けられたのか」が直感的に分かりにくい状態でした。

解決：

攻撃後に一定時間のディレイを入れて硬直を作り、さらに「威嚇行動」と「攻撃」を確率でランダムに選択するようにして、常に攻撃し続けたい挙動に変更しました。あわせて、回避成功時には背景を暗くし、敵の動きをスローにして無敵状態になる「チャンスタイム」を実装し、視覚的にも体感的にも分かりやすいご褒美を用意しました。

効果：

敵に明確な隙と威嚇時間が生まれたことで、プレイヤーから反撃しやすくなり、戦闘のテンポにメリハリが付きました。チャンスタイムでは「避けた瞬間に一気に攻め込める」という爽快感が生まれ、防御行動が単なるやり過ごしではなく、積極的に狙いたくなる要素になりました。

学び：

敵 AI は「強くする」だけでなく、プレイヤーが攻めに転じられる“間”をどう設計するかが重要だと実感しました。また、防御アクションには、成功時に分かりやすいフィードバックと明確なリターンを与えることで、初めて楽しい要素になることを学びました。

### 課題③：飛行アクションの影響でステージ設計が単調になってしまったこと

課題：

飛行を実装したことで、ステージの意図したルートが崩れやすくなりました。同じ高さに足場を並べた箇所では、本来は1→2→3と進ませたいところを、飛行で2を飛ばして3に直接到達できてしまい、切り替え床もオフの床を無視してオンの床まで飛んでしまえばギミックを無視できる状態でした。その結果、ステージ攻略が想定より簡単になり、単調な印象になっていました。

解決：

足場の配置を横並びから縦方向中心に見直し、高さの違う足場を組み合わせることで、一度の飛行で到達できる範囲を明確に制限しました。また、切り替え床は「飛行開始時」と「任意のタイミングで飛行を終了したとき」にオン／オフが切り替わる仕様に変更し、「どのタイミングで飛行を開始・終了するか」を考えさせる設計にしました。

効果：

高低差のある足場が増えたことで、プレイヤーから見ても進むべき方向が視覚的に分かりやすくなり、大きなショートカットルートもほぼ解消されました。また、切り替え床のトグルタイミングを意識する必要性が生まれ、飛行とギミックの組み合わせを考えながら進む、パズル性のあるステージ攻略になりました。

学び：

強力な移動手段を導入するときは、その挙動の限界（距離や高さ）を踏まえてステージ側の設計もセットで見直す必要があると学びました。ギミックは単に置くだ

けではなく、「アクションのどのタイミングと連動させるか」まで設計することで、プレイヤーに自然な思考と達成感を与えられると実感しました。

## 5. まとめ

### 本作品で達成できたこと

- ・「飛行」をコアアクションとして実装し、移動床・トランポリン・トグル床などのステージギミックと組み合わせることで、いつ飛ぶかを考えながら進む立体的なステージ攻略を実現しました。

- ・敵の攻撃に対するジャスト回避から「チャンスタイム」に突入する戦闘システムを構築し、通常時の緊張感ある立ち回りと、チャンスタイム中のスロー演出＋ラッシュ攻撃による爽快感を両立させた 3D アクションとしてまとめ上げました。

- ・Unreal Engine 5 上で Blueprint を用いて、PlayerCharacter／EnemyBase／GameMode／GameInstance／SaveGame／各種 Widget などの責務を分離しながら、飛行アクション・ステージギミック・クエスト進行・UI を含む 3D アクションゲーム全体を一人で設計・実装し、プレイアブルな作品として形にしました。

### 今後の展望

- ・本作の特徴である飛行アクションをより活かせるよう、飛行と連動したギミックを用いたステージバリエーションを増やし、ゲームボリュームと遊びの幅をさらに広げていきたいと考えています。

- ・仲間キャラクターの AI を実装したり、ローカル／オンライン協力プレイに対応させたりすることで、誰かと共闘しながらステージを攻略する協力プレイ体験を実現したいと考えています。

- ・モンスターから素材を獲得し、装備の作成や装備による特殊スキルの発動が行える仕組みを導入することで、同じクエストを繰り返し遊べるリプレイ性と、スキル構成を工夫するビルドの楽しさ・快適さを提供していきたいです。