

# HackGunner 説明資料

東京工芸大学 芸術学部 ゲーム学科 鈴木十音

# 目次

- HackGunnerとは
- 概要
- 担当箇所
- チーム内での役割（全体で）
- チーム内での役割（プログラマーとして）
- 工夫点
- プログラム例:「CharacterStatus.cs」の仕組み
- プログラム例:「GunStatus.cs」の仕組み
- ソースコード

# HackGunnerとは

- コンセプト：倒した敵に乗り移って敵を全滅させるFPS
- ターゲット：コウゲイゲームショウ2025に遊びにくる中・高・大学生



敵を倒して...



乗り移る！

# 概要 (1/2)

- タイトル

「HackGunner」

- コンセプト

倒した敵に乗り移って敵を全滅させるFPS

- 目的

コウゲイゲームショウ2025への出展

# 概要 (2/2)

## ○ 開発期間

10ヶ月(2024年4月～2025年1月)

## ○ 開発環境

Unity(2022.3.25f1)

Visual Studio 2022

Visual Studio Code

GitHub

Miro

## ○ メンバー

プランナー : 2人

デザイナー : 2人

プログラマー : 4人 (本人含む)

# 担当箇所

## ○ キャラクター共通

- ・ ステータスのデータ用クラス
- ・ キャラクターの状態（例：乗り移れるか）
- ・ キャラクターの被弾処理
- ・ キャラクターの撃破処理
- ・ ステータスの切り替え

## ○ 弾丸

- ・ 進捗処理
- ・ 衝突判定
- ・ 衝突処理

## ○ データクラス

- ・ キャラクター（プレイヤー・敵）
- ・ 銃
- ・ 弾丸

## ○ UI

- ・ 使用中の武器アイコン表示
- ・ 敵残数の表示
- ・ スロー状態、HP減少時の画面エフェクト

# チーム内での役割（全体で）

## ○ チーム内の橋渡し

- ・ 提案や指示された仕様とコンセプトとの食い違いを整理し共有した
- ・ チームメンバーがそれぞれの作業に必要としている資料や情報を整理し、スムーズに制作できるようサポートした

例）プランナーとデザイナーの間にイメージの共有が難航した際、プランナーの意図を言語化して参考資料をまとめ、それをデザイナーに共有したことで制作が再開できた。

- ・ 意見が衝突したとき、それぞれの意見のどこが食い違っているのかを見つけ出し解決へと導いた

# チーム内での役割（プログラマーとして）

## ○ タスクの割り振り

- ・ 直近のタスク等から他のメンバー（主にプログラマーに対してだが、プランナーやデザイナーも含む）の作業の適性を考慮し、プランナーが割り振りきれなかったタスクを適任者に再配分した。

また、プランナーがタスクを割り振る際には、適任者を理由を共有しながら一緒に検討することで、納得感のある体制づくりを支援した

- ・ 他のプログラマーのメンバーと仕様を確認し合う際、実装をイメージしやすい形で仕様を言語化し、認識の齟齬を防いだ

→ チーム全体の開発効率を高め、メンバーが納得して作業できる環境づくりに努めた。



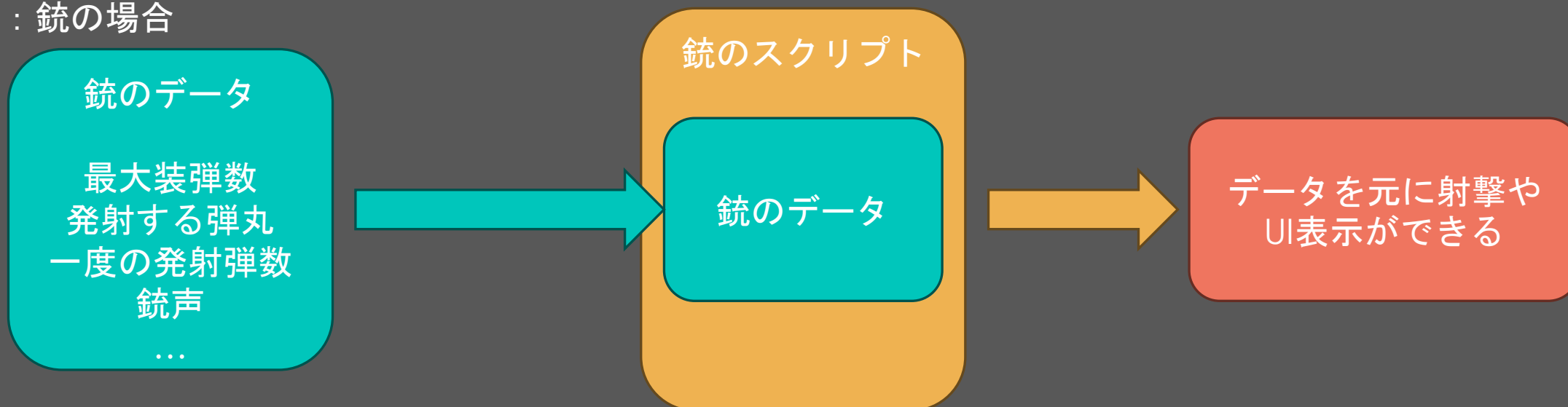
# 工夫点 (1/3)

課題：銃の種類を変更するために、銃ごとにプレハブを切り替えるのは手間がかかる

キャラクターや弾丸、銃のデータ設定をわかりやすくしたい

→ ScriptableObjectを用いたデータの設定をもつクラスを作成し、スクリプトにアタッチするだけで登録できるようにして、設定する箇所を分かりやすくできた！

例：銃の場合



# 工夫点 (2/3)

課題：乗り移ることを楽しんでほしい

→チーム内で乗り移るときに起きるイイことや乗り移りのテンポ感についての仕様を話し合い、実際に動きを見ながら実装した！

（イイことの例：乗り移ると体力回復、倒した敵の位置に移動できる）

（テンポ感の例：乗り移り時に頭が回る動き）



体力の回復



相手の武器を使用

# 工夫点 (3/3)

課題：乗り移りをしているかどうか分かりづらい

→ チーム内で乗り移るときの動きのアイデアを出し合い、ひと目で乗り移っているとわかるような動きを作成した！

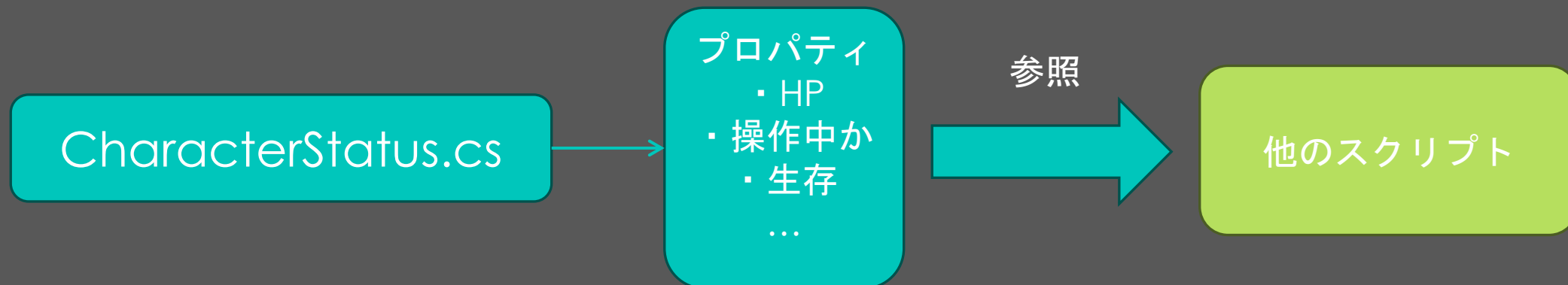
（例：頭を回転させながら投げ飛ばす、投げている間はカメラを引いて三人称視点に）

乗り移りをしていることがわかりやすくなったことで、プレイ中のストレスを軽減することができた。



# プログラム例:「CharacterStatus.cs」の仕組み

- 最大HPや被弾時の無敵時間といったステータスのデータ用のクラスを作成している (CharacterData型)
- 敵のときとプレイヤー操作時にステータスを分ける必要があったため、それぞれあらかじめ設定している
- 現在HPやプレイヤーが操作している状態かといった情報を、他のスクリプトからでも参照できるようにプロパティ化している
- スクリプト内でのメソッドは、ダメージを受ける処理や乗り移られるときの処理等を行っている



# プログラム例: 「GunStatus.cs」の仕組み

- 発射する弾丸の種類や拡散、サブ武器を持つかといったデータ用のクラスを作成している (CharacterData型)
- アイコンや残弾数等のUIの表示で使用するものをプロパティ化している (他のスクリプトから取りやすいように)
- スクリプト内でのメソッドは、銃の発射やサブ武器への切り替え処理を行っている



# ソースコード (1/4)

## Deta部分

スクリプト名	概要
<a href="#">BaseDeta.cs</a>	他のデータに共通する変数を持つ、 他のデータクラスの継承元
<a href="#">BulletData.cs</a>	弾丸のデータ
<a href="#">CharacterData.cs</a>	プレイヤーや敵等のキャラクターのデータ
<a href="#">WeaponData.cs</a>	武器や発射する弾丸の種類のデータ

各スクリプト名のリンクをクリックでGitHub  
のページへと飛びます

# ソースコード (2/4)

## ゲーム内動作

スクリプト名	概要
<a href="#">PlayerHeadManager.cs</a>	プレイヤーの頭の有効無効を切り替える
<a href="#">PlayerHeadMoveScript.cs</a>	投げたプレイヤーの頭を動かす
<a href="#">CharacterStatus.cs</a>	キャラクターのステータスや動作の管理
<a href="#">GunStatus.cs</a>	銃の種類管理、発砲を行う
<a href="#">EnemyHeadBlowScript.cs</a>	倒した敵の頭を飛ばす

各スクリプト名のリンクをクリックでGitHub  
のページへと飛びます

# ソースコード (3/4)

## ゲーム内動作、他の人との共同

スクリプト名	概要/担当箇所
<a href="#">Change.cs</a>	乗り移り動作/頭が到着してから切り替えるようにした
<a href="#">PlayerMove.cs</a>	プレイヤーの動作を行う/持っている武器の更新と歩行音の再生を担当
<a href="#">PlayerRay.cs</a>	視点や乗り移り、射撃の処理/乗り移りの取得範囲拡大と射撃処理を担当
<a href="#">BulletBaseClass.cs</a>	弾丸の動作/弾丸の移動と衝突処理を担当

各スクリプト名のリンクをクリックでGitHub  
のページへと飛びます



# ソースコード (4/4)

## UI部分

スクリプト名	概要
<a href="#">GunUIScript.cs</a>	武器のUI表示
<a href="#">RemainEnemyNumScript.cs</a>	ステージ上の敵の残数を表示
<a href="#">ScreenEffectScript.cs</a>	スロー状態の画面エフェクト

各スクリプト名のリンクをクリックでGitHub  
のページへと飛びます