

ENERGY PANDEMIC

解説ドキュメント

MAI ZHICONG
(バク チソウ)






日本電子専門学校 ゲーム制作研究科

TEL : 080-9370-9449
Mail : 23cu0226@jec.ac.jp

作品概要

ジャンル	2D アクションゲーム
作成環境	Unity2022.3.22f1・Visual Studio Code
作成期間	2ヶ月（2024年3月～4月, 2024年7月～8月）
作成メンバー数	6人（プランナー1人、デザイナー1人、プログラマー4人）

実行概要

実行環境	PC (Windows11)
実行手順	<div> baselib.dll  GameAssembly.dll  UnityCrashHandler64.exe  UnityPlayer.dll  Zombie.exe</div> <p>実行ファイル/Zombie.exe を起動</p>

フォルダー構成

成果物リスト	フォルダパス
操作説明書	実行ファイル/操作説明
実行ファイル	実行ファイル/Zombie.exe
プロジェクトデータ	プロジェクトファイル.zip

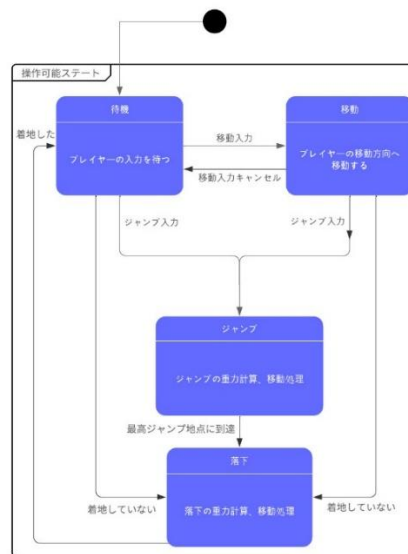
作品担当箇所

<u>プレイヤー制御</u>
<u>エネルギー制御（メインシステム）</u>
<u>シェーダー作成</u>
中間ポイント作成
UI 制御
パーティクル制御
オープニング、シナリオ制御
外部ファイル操作制御
データ設計、実装
音声再生オブジェクト制御

担当箇所解説

- ・ プレイヤー制御 :
 - └ プレイヤーステートマシン

プレイヤーアクションステートマシン(一部)



工夫した点 :

状態でプレイヤーの管理することによって、大量の if 文制御を避けることができ、各状態で特定の処理を集中させ、処理の修正・新しいロジックの追加がしやすくなります。

該当ソースコード :

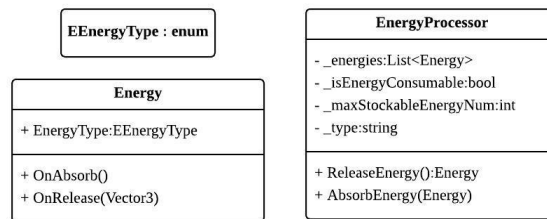
- ・ ステートマシン雛形 : Assets/Utilities/StateMachine
- ・ プレイヤー専用 :
Assets/Scripts/Character/Player/PlayerActionState

・ エナジー制御システム :

└ エナジーシステム構成:

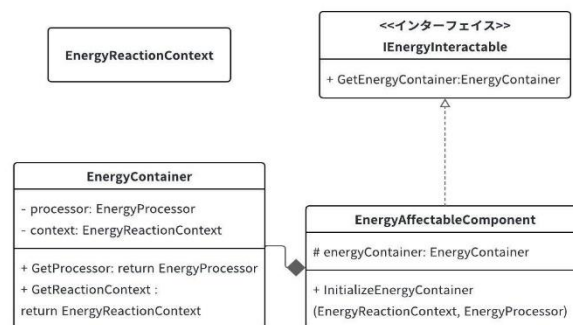
① エナジープロセッサ :

エナジーの吸収・リリース処理、エナジーインスタントの管理。



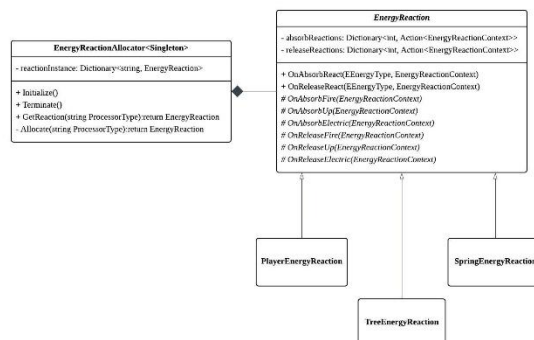
② エナジーコンテナ :

エナジープロセッサ、エナジーリアクションコンテキストを管理

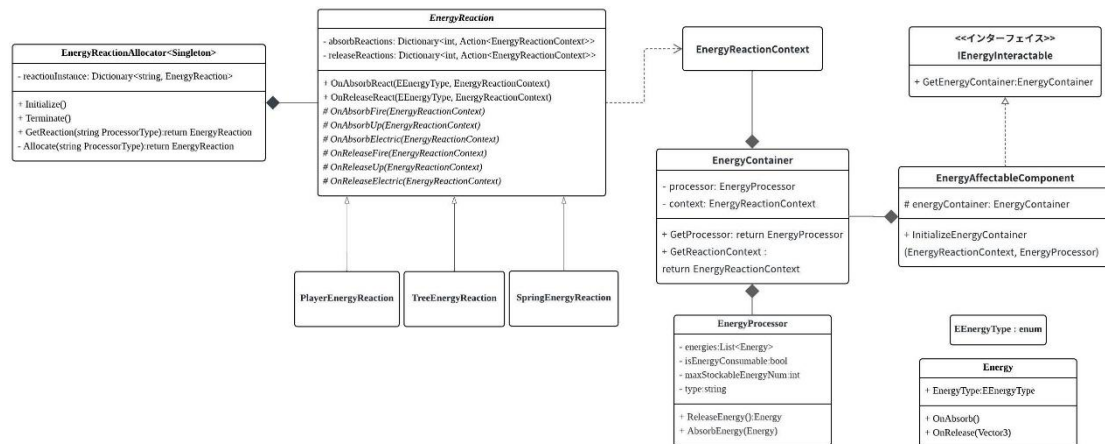


③ エナジーリアクション :

エナジーの吸収に対して反応するイベントハンドラ



システム全貌：



工夫した点：

それぞれのクラスに一つの責任を持たせることを意識し、新しいオブジェクトリアクションを追加する時に既存の親クラスから継承して作るだけで拡張が簡単にできます。

該当ソースコード：

- ・ システムコア：Assets/Scripts/Energy/Core
- ・ 具体実装：Assets/Scripts/Energy/EnergyReaction

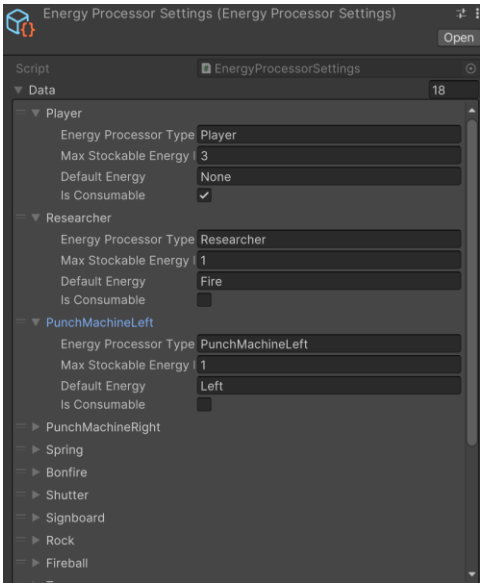
└ エナジープロセッサの初期化：

外部 Excel ファイルにデータを入れ、データを読み込み、Json 形式で Unity の ScriptableObject に書き込みます。ScriptableObject を利用して、プロセッサの設定オブジェクトを初期化します。

外部ファイル：

EnergyProcessorType	MaxStockableEnergyNum	DefaultEnergy	IsConsumable
Player	3	None	TRUE
Researcher	1	Fire	FALSE
PunchMachineLeft	1	Left	FALSE
PunchMachineRight	1	Right	FALSE
Spring	1	Up	FALSE
Bonfire	1	Fire	FALSE
Shutter	0	None	FALSE
Signboard	0	None	FALSE
Rock	1	Down	FALSE
Fireball	1	Fire	FALSE
Tree	0	None	FALSE
ElectricLine	1	Electric	FALSE
None	0	None	FALSE
Cheater	9999	None	FALSE
Doctor	42	Fire	FALSE
Bomb	1	None	FALSE
MovingFloor	0	None	FALSE
ConveyorBelt	0	None	FALSE

Unity Scriptable Object：



工夫した点：

設定データを外部ファイルに保存して、エンジンがファイルを読み込んでゲームデータに書き込みます。エンジニア以外の人もデータを修正することができ、エンジン側でデータの有効性を確認することができます。

該当ソースコード：

外部ファイル読み込み：Assets/Editor/FileImportSystem

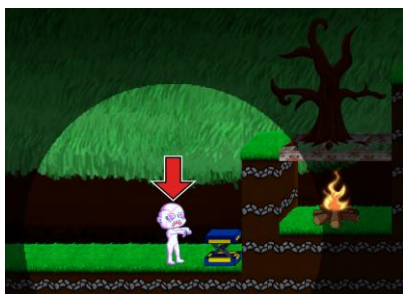
Jsonに変換：Assets/Editor/ConvertToJsonHelper

エナジープロセッサ設定リポジトリ：

Assets/Scripts/Energy/EnergyProcessorSetting

・シェーダー作成

エナジー選択モードのステンシルバッファ：



範囲外あるいはエナジーリリースターゲットじゃないオブジェクトにフォグをかける。

該当ソースコード：

Assets/Resources/Shader/DimEffect.shader

Assets/Resources/Shader/FogUnaffectedShader.shader

アウトライン：

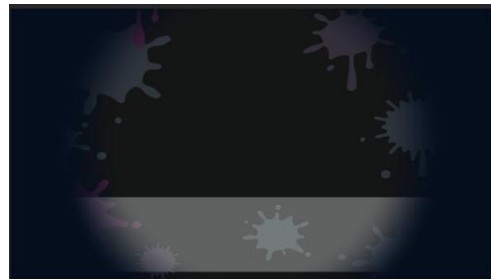


Unity の Outline コンポーネントだときれいなアウトラインが出ないため、自作シェーダーでアウトラインを作成しました。

該当ソースコード：

Assets/Resources/Shader/OutlineEffect.shader

シーン切り替えのエフェクト：



シーンを切り替える時、フェードインとフェードアウト用のエフェクトをシェーダーで作成しました。

該当ソースコード：

シェーダー：Assets/Resources/Shader/FadeInOut.shader

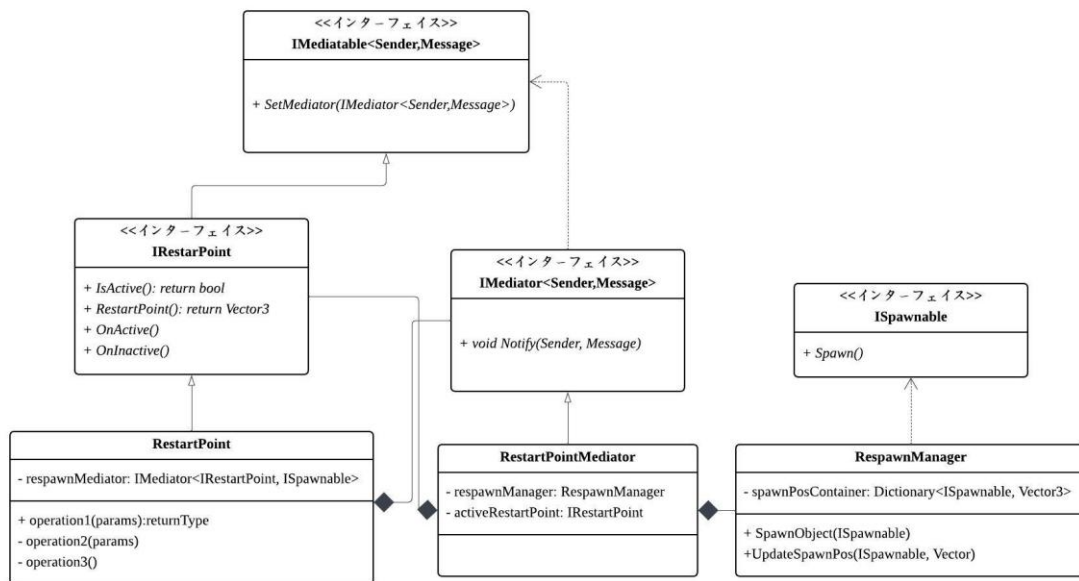
C#使用：Assets/Scripts/Effect/SwitchSceneEffect

データモデル：Assets/Scripts/Model/FadeInOutModel.cs

・他項目説明：

① 中間ポイント：

プレイヤーが中間ポイントを触ると、中間ポイントが仲介を経由しマネージャーにイベント通知を送ります。



該当ソースコード：

中間ポイント：Assets/Scripts/RestartPoint

再生マネージャー：Assets/Scripts/Managers/RespawnManager.cs

② UI 制御：

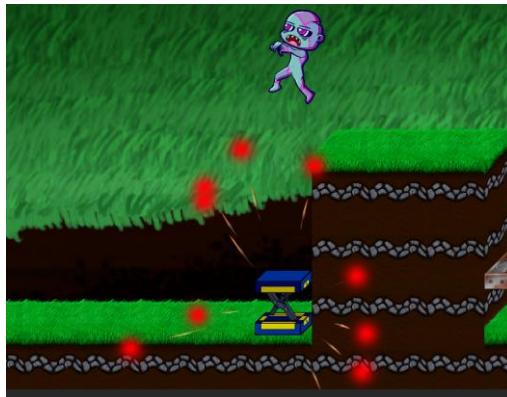
プレイヤーロジック部分を直接 UI に触らず、オブザーバーを経由して、UI の更新を行います。

③ パーティクル制御：

エナジーを別のオブジェクトにリリースする際に、パーティクルでエナジーの流れを表せます。制御は二段階に分けて計算して処理を行っています。

第一フェース：パーティクルがリリース発生源から外へ拡散

例：プレイヤーがばねを踏んだ時



パーティクルが直線移動をし、どんどんスピードを落とします。

第二フェース：パーティクルがターゲットに向けて移動

例：プレイヤーがばねからエネルギーを獲得する時



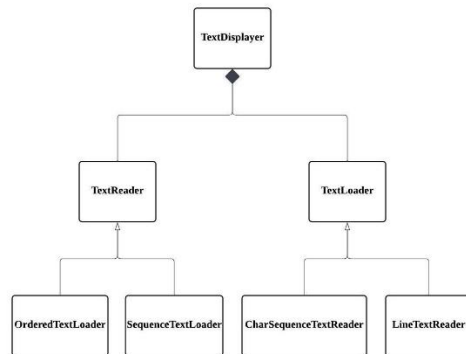
パーティクルのスピードの方向とパーティクルからターゲットまでのベクトルの角度（ラジアン）を計算し、スピードの方向を修正していきます。

該当ソースコード：

Assets/Scripts/Effect/EnergyEffect/EnergyTransformEffect.cs

④ オープニング、シナリオ制御

シナリオテキストの読み込みと書き出し制御を分けて作成しました



該当ソースコード：

Assets/Scripts/TextDisplayer

⑤ 外部ファイル操作制御

シナリオ専用のフォーマットを作成し、専用の拡張子を Unity に識別させ、読み込んでデータオブジェクトに書き込みます。

拡張子：.ztf

フォーマット：

```
<FileType>Text</FileType>
<ID>10001</ID>
時は 2 0 X X 年ーー。
人類は滅びようとしていた。
<ID>10002</ID>
パラソル社が開発した新型ウイルス
通称「ゾンビウイルス」
これに起因する世界規模で発生した
パンデミックが原因である。
<ID>10003</ID>
そんな世界でも危険な研究を
続けている一人の博士がいた。
```

該当ソースコード：

ファイル操作：

`Assets/Editor/FileImportSystem/ZtfImportSystem.cs`

`Assets/Editor/ZAssetPostProcessor/ ZAssetPostProcessor.cs`

ファイル：

`Assets/Resources/ImportZtfFile/Opening.ztf`

⑥ 音声再生オブジェクト制御：

AudioSource を搭載した発声オブジェクトをプールで管理し、効果音を鳴らす時、プールからオブジェクトを取り出し、してされた座標に配置し、音を鳴らします。鳴らし終わったら、自動的にプールに戻ります。プールで管理すると、頻繁にメモリを再確保（new）することによる処理負荷を無くすことができます。

該当ソースコード：

オブジェクトプール雛形：`Assets/Utilities/Pool`

音声管理マネージャー：`Assets/Scripts/Managers/AudioManager`