PlayerController

スボタンがおされたなら

attack アニメ(トリガー)

Player	Lontroller				
private public	float Rigidbody2D Animator GameObject	axisH axisV angleZ rbody anime	//Player 移動スピード // 水平方向の入力値 // 垂直方向の入力値 // 向いている角度 //Rigidbody2D コンポーネント //Animator コンポーネント // ライトの子オブジェクト // ダメージ中フラグ		
	Start				GetDamage
コンポー	ネント取得			gai	meState が playing でないと return
SpotLigh	ntCheck メソッド			プ	レイヤー HP を減らす
Animatic Attack >	e メソッドで angleZ on メソッド メソッド ixedUpdate ate が playing でない	≥ return	現在値と予測方向の算出 水平か垂直のキー入力があれば あらためて角度を算出 1フレーム前の角度を維持 算出した角度を return Animation	isD	HPがあれば ノックバック DamageEnd メソッド GameOver メソッド DamageEnd Oamage フラグを false
	メージ処理して return で動かす		水平か垂直のキー入力があれば Run アニメあり(ブール) 各アングルによって分岐 direction アニメ(Integer)	Pla	ayer を確実に表示
hasSpot	otLightCheck Light なら otLight を表示		左右の向きを転換 Run アニメなし(ブール)	_	GameOver meState を gameover に ームオーバー演出

OnCollisionEnter2D

GetDamage メソッド

相手が Enemy なら

Shooter

```
private PlayerController playerCnt //PlayerController コンポーネント public float shootSpeed // 投擲物のスピード public float shootDelay // 発射間隔 private bool inAttack // 攻撃中フラグ
```



SpotLight

private PlayerController playerCnt //PlayerController コンポーネント public float rotationSpeed // 回転速度



GameManager

public static GameState gameState // ゲームのステータス

public static bool doorsOenedState // ドアの開閉状況

public static int key1 // 鍵 1 の持ち数 public static int key2 // 鍵 2 の持ち数 public static int key3 // 鍵 3 の持ち数

public static bool[] keysPickedState // 鍵の取得状況

public static int bill // お札の持ち数 public static bool[] itemsPickedState// アイテムの取得況

public static bool hasSpotLight // スポットライトを持っているか public static int playerHP // プレイヤーの HP

public enum GameState

- playing
- talk
- gameover
- gameclear
- stop
- ending

[Player]

(特徴)

Player は上下左右でアニメを切り替えながら移動ができます。

スペースキーでお札を投擲できます(Shooter.cs)

子オブジェクトに SpotLight がいて、プレイヤーにあわせて方向転換します(SpotLight.cs)

Player の HP は GameManager が管理し、ダメージをもらうと点滅しながらノックバックします

