Documentación del Patrón de Diseño: State (Estado)

1. El patrón en general (definición, propósito)

El patrón de diseño **State (Estado)** es un patrón de comportamiento que permite que un objeto modifique su comportamiento cuando cambia su estado interno. El objeto parecerá cambiar su clase.

Este patrón ayuda a evitar largas estructuras condicionales, encapsulando el comportamiento de cada estado en una estructura diferente, lo que mejora la modularidad y el mantenimiento del código.

2. Cómo fue implementado en el proyecto

En este proyecto se aplicó el patrón State en el **jugador**, quien puede encontrarse en distintos estados según los objetos que recolecta.

Se definió un enum llamado PlayerState con los siguientes posibles estados:

```
enum PlayerState { NORMAL, CON_LINTERNA, CON_MACHETE }
var state: PlayerState = PlayerState.NORMAL
```

El estado del jugador cambia dinámicamente cuando colisiona con objetos como la linterna o el machete. Por ejemplo:

```
# En el script de la linterna
func _on_body_entered(body):
	if body is CharacterBody2D:
		if item_type == "linterna":
			body.state = body.PlayerState.CON_LINTERNA
```

En el script del jugador, el estado es mostrado en pantalla y puede usarse para modificar su comportamiento:

```
$Label.text = "Estado: " + get_estado_nombre()

func get_estado_nombre():
	match state:
		PlayerState.NORMAL:
			return "NORMAL"
		PlayerState.CON_LINTERNA:
```

```
                return "CON_LINTERNA"
            PlayerState.CON_MACHETE:
                return "CON_MACHETE"
```

3. Justificación de su uso en el contexto del sistema desarrollado

Esta implementación permite que el jugador tenga distintos comportamientos o interacciones según su estado.

Por ejemplo:

-Si tiene la linterna, más adelante puede iluminar zonas oscuras.
-Si tiene el machete, puede cortar obstáculos.

Este patrón ayuda a que el código del jugador sea claro y extensible, ya que cada nuevo estado puede agregarse fácilmente al enum y a la lógica asociada.

4. El uso del patrón debe estar reflejado claramente en el código fuente

El patrón está reflejado en el uso del enum PlayerState, el cual define los posibles estados del jugador. El cambio de estado se realiza dinámicamente en tiempo de ejecución, y se refleja en la interfaz del juego en el estado de prueba en el que nos encontramos, sin embargo en la versión final del juego el jugador podrá identificar el estado del personaje de otra manera .

Esto permite que el comportamiento del jugador se adapte según el contexto, tal como define el patrón de diseño **State**.