API de Managers

Documentación de los managers Autoload del sistema de energía y grid.

GridManager

Ruta: `scripts/managers/grid_manager.gd`
Autoload: `GridManager`

Registra qué celdas del grid están ocupadas por edificios. Usado por `ConstructionManager` para validar colocación y por `SaveSystem` al reconstruir.

Constantes

Constante	Tipo	Valor	Descripción
`GRID_SIZE`	float	1.0	Tamaño de celda del grid

Funciones

`limpiar() -> void`

Limpia todo el registro. **Llamar al iniciar o cargar nueva partida.**

`register_building(pos: Vector2i, building) -> void`

Registra un edificio en una posición del grid.

- **pos**: coordenadas (x, z) de la celda
- **building**: nodo del edificio

`unregister_building(pos: Vector2i) -> void`

Elimina el registro de un edificio en una posición.

`unregister_building_all(building) -> void`

Elimina **todas** las celdas que ocupa un edificio (para edificios multi-celda como el Merger 3×1). Usado al levantar el edificio del suelo.

`is_cell_occupied(pos: Vector2i) -> bool`

Indica si una celda está ocupada.

`get_building_at(pos: Vector2i) -> Node`

Devuelve el edificio en esa celda, o `null` si está vacía.

`hay_edificio_en_radio(centro: Vector2i, radio: int) -> bool`

Comprueba si hay algún edificio en el cuadrado `centro ± radio`. Usado para validar colocación (evitar solapamientos).

- **centro**: celda central
- **radio**: casillas en cada dirección (0 = solo centro, 1 = 3×3)

EnergyManager

Ruta: `scripts/managers/energy_manager.gd`
Autoload: `EnergyManager`

Gestiona los flujos de energía numéricos. Crea, actualiza y elimina `EnergyFlow` y opcionalmente muestra visuales de pulsos.

Señales

Señal	Parámetros	Cuándo se emite
`energy_transferred`	`from_building: Node`, `to_building: Node`, `amount: int`	Cuando un flujo termina y entrega la energía

Constantes

Constante	Tipo	Valor	Descripción
`MOSTRAR_VISUAL_PULSO`	bool	true	Activar/desactivar esferas visuales de energía
`FLUJO_DURACION_BASE`	float	0.5	Duración mínima si distancia ≈ 0

Funciones

`register_flow(from: Node, to: Node, amount: int, tipo_recurso: String, color: Color, duration: float = -1.0) -> EnergyFlow`

Registra un flujo de energía entre dos edificios.

- **from**: edificio emisor
- **to**: edificio receptor (debe tener `recibir_energia_numerica`)
- **amount**: cantidad de energía
- **tipo_recurso**: ej. "Stability", "Charge", "Compressed-Stability", "Up-Quark", etc.
- **color**: color del visual
- **duration**: duración en segundos; si es -1 se calcula por distancia
- **Retorna**: `EnergyFlow` o `null` si algún edificio es inválido

`spawn_pulse_visual(from_pos: Vector3, to_pos: Vector3, color: Color, source_origen: Node = null) -> void`

Crea el visual de la bola de energía (solo si `MOSTRAR_VISUAL_PULSO = true`).

- **from_pos**, **to_pos**: posiciones mundo
- **source_origen**: si se pasa y rota, el visual se destruye para no flotar en el aire

`unregister_flow(flow: EnergyFlow) -> void`

Elimina un flujo del registro (usado internamente al completar).

EnergyFlow

Ruta: `scripts/managers/energy_flow.gd`
Tipo: `RefCounted` (clase de datos, no nodo)

Representa un flujo de energía en tránsito. No se instancia manualmente; lo crea `EnergyManager.register_flow()`.

Propiedades

Propiedad	Tipo	Descripción
`from_building`	Node	Edificio emisor
`to_building`	Node	Edificio receptor
`amount`	int	Cantidad de energía
`tipo_recurso`	String	Tipo de recurso
`color`	Color	Color del visual
`duration`	float	Duración total en segundos
`timer`	float	Tiempo transcurrido (interno)

Funciones

`update(delta: float) -> bool`

Actualiza el flujo. Llamado por `EnergyManager` cada frame.

- **Retorna**: `true` si sigue activo, `false` si ya entregó (y debe eliminarse)

Al completar, llama a `to_building.recibir_energia_numerica(amount, tipo_recurso, from_building)`.

BuildingManager

Ruta: `scripts/managers/building_manager.gd`
Autoload: `BuildingManager`

Lista de edificios activos. Usado por Siphons, Compressors, etc. para saber qué edificios existen.

Funciones

`limpiar() -> void`

Vacía la lista. **Llamar al iniciar o cargar nueva partida.**

`register_building(building: Node) -> void`

Añade un edificio a la lista.

- **building**: nodo del edificio (Area3D con lógica de edificio)

`unregister_building(building: Node) -> void`

Quita un edificio de la lista.

`get_buildings_in_radius(pos: Vector3, radius: float) -> Array`

Devuelve todos los edificios dentro del radio dado.

- **pos**: posición mundo (Vector3)

- **radius**: radio en unidades
- **Retorna**: Array de nodos

Método obligatorio para receptores de energía

Cualquier edificio que pueda recibir energía debe implementar:

```
func recibir_energia_numerica(cantidad: int, tipo_recurso: String, origen: Node) -> void
```

Implementado en: Compressor, Merger, Prism, Constructor.