📋 Lista de Tareas de Programación - Juego Stealth 3D

🎮 PERSONAJE JUGABLE

A. Movimiento Básico (PRIORIDAD ALTA - BASE DEL JUEGO)

- [ ] A1 - Sistema de movimiento WASD con CharacterController
  - Variables públicas: `velocidadCaminar`, `velocidadCorrer`
  - Detección de input para caminar/correr
- [ ] A2 - Sistema de agachado
  - Toggle con tecla (Ctrl/C)
  - Reducir altura del collider cuando está agachado
  - Variable pública: `velocidadAgachado`
- [ ] A3 - Sistema de pegado a pared
  - Raycast para detectar paredes cercanas
  - Alineación del personaje a la pared
  - Movimiento lateral sobre la pared

Conexiones:

- Se conecta con B1-B4 (Cámara)
- Se conecta con G1-G2 (Detección de enemigos por sonido)
- Se conecta con L3 (Audio de pasos)

B. Sistema de Cámara (PRIORIDAD ALTA)

- [ ] B1 - Cámara base tercera persona con Cinemachine
  - Follow y LookAt al jugador
  - Variables públicas: `sensibilidadX`, `sensibilidadY`, `distanciaCamara`
- [ ] B2 - Shake de cámara por eventos (screamers)
  - Variable pública: `intensidadVibracion`
  - Función: `ShakeCamara(float duracion, float intensidad)`
- [ ] B3 - Shake de cámara por movimiento (caminar/correr)
  - Variables públicas: `shakeCaminar`, `shakeCorrer`
  - Amplitud y frecuencia configurables
- [ ] B4 - Zoom dinámico al agacharse
  - Reducir FOV y acercar cámara al agacharse
  - Transición suave con Lerp
  - Variable pública: `fovAgachado`
- [ ] B5 - Zoom de apuntar (sobre el hombro)
  - Cambio de posición de cámara al hombro
  - Activar crosshair/mira
  - Variables públicas: `fovApuntar`, `offsetHombroX`, `offsetHombroY`
- [ ] B6 - Zoom de exploración
  - Zoom in/out libre con scroll o tecla
  - Variables públicas: `fovMinExplorar`, `fovMaxExplorar`, `velocidadZoom`
- [ ] B7 - Sistema de acercamiento cuando enemigo cerca (escondido)
  - Detectar proximidad de enemigos cuando está escondido
  - Mini shake y acercamiento progresivo
  - Variables públicas: `rangoDeteccionEnemigo`, `intensidadShakeProximidad`

Conexiones:

- Se conecta con A1-A3 (Movimiento)
- Se conecta con C1-C3 (Habilidades para zoom en menú)
- Se conecta con H1 (NPC screamers)
- Se conecta con J2 (Menú inventario/habilidades)
- Se conecta con M1 (Configuración sensibilidad)

C. Sistema de Habilidades (PRIORIDAD MEDIA-ALTA)

- [ ] C1 - Habilidad: Onda Electromagnética
  - Detección de enemigos en rango con OverlapSphere
  - Aplicar stun a enemigos detectados
  - Variables públicas: `rangoOnda`, `duracionStun`, `cooldownOnda`
  - Visualización de rango con Gizmos (color configurable)
  - Cooldown timer
- [ ] C2 - Habilidad: Ralentización de Tiempo
  - Modificar Time.timeScale para ralentizar
  - Excluir al jugador del efecto (usar FixedUpdate sin timeScale)
  - Variables públicas: `factorRalentizacion`, `duracionRalentizacion`, `cooldownRalentizacion`
  - Cooldown timer
- [ ] C3 - Habilidad: Control de Drones
  - Sistema de disparo/selección de enemigo objetivo
  - Cambiar facción del enemigo temporalmente
  - Hacer que ataque a enemigos cercanos
  - Variables públicas: `duracionControl`, `cooldownControlDrones`, `rangoDeteccionAliados`
  - Cooldown timer
- [ ] C4 - Selector de habilidades (input)
  - Teclas 1, 2, 3 para seleccionar habilidad activa
  - Indicador visual de habilidad seleccionada

Conexiones:

- Se conecta con G1-G6 (Estados de enemigos - stun, control)
- Se conecta con J2 (UI selector de habilidades)
- Se conecta con K3 (HUD cooldowns estilo Dead Space)
- Se conecta con L2 (Audio Manager para SFX de habilidades)

D. Sistema de Salud y Daño (PRIORIDAD ALTA)

- [ ] D1 - Script de vida del jugador
  - Variable pública: `vidaMaxima`, `vidaActual`
  - Función: `RecibirDanio(float cantidad)`
- [ ] D2 - Integración con Post-Processing de daño
  - Llamar a `PostProcessManager.SetDamageVignette()` al recibir daño
  - Intensidad de vignette basada en vida restante
- [ ] D3 - Sistema de muerte y respawn
  - Detectar vida <= 0
  - Llamar a `PostProcessManager.ActivarVignetteMuerte()`
  - Trigger del Quick Time Event
- [ ] D4 - Integración Quick Time Event (QTE)
  - Conexión con tu sistema de QTE existente
  - Si falla: reinicio desde checkpoint
  - Si acierta: llamar a `PostProcessManager.LerpVignetteNormal()` para restaurar
  - Variable pública: `tiempoMaximoQTE`

Conexiones:

- Se conecta con F1 (Explosivos)
- Se conecta con I1-I2 (Sistema de checkpoints)
- Se conecta con N1 (PostProcessManager - NUEVO)
- Se conecta con L2 (Audio de daño)

E. Sistema de Interacción (PRIORIDAD MEDIA)

- [ ] E1 - Detección de objetos interactuables
  - Raycast desde cámara o esfera de detección
  - Variable pública: `rangoInteraccion`
  - Mostrar prompt de interacción (llamar a InteractionUI)
- [ ] E2 - Interacción con NPCs estáticos
  - Trigger de diálogos/screamers
- [ ] E3 - Recolección de objetos
  - Tarjetas de acceso (llaves)
  - Objetos especiales (cinemáticas)
  - Añadir al inventario al recolectar
  - Llamar a `InventoryUI.AgregarItem()`
- [ ] E4 - Sistema de shader de brillo para objetos importantes
  - Shader con emisión configurable
  - Intensidad basada en distancia al jugador
  - Variables públicas: `distanciaMaxBrillo`, `intensidadMaxBrillo`, `distanciaMinBrillo`

Conexiones:

- Se conecta con H1-H2 (NPCs)
- Se conecta con J1 (InventoryUI)
- Se conecta con K2 (InteractionUI)

👹 ENEMIGOS / NPCs

F. Enemigos - Explosivos Estáticos (PRIORIDAD BAJA - FÁCIL)

- [ ] F1 - Enemigo explosivo con movimiento punto a punto
  - Lista pública: `List<GameObject> puntosPatrulla`
  - Variable pública: `velocidadMovimiento`
  - Movimiento lineal entre puntos
  - Detección de colisión con jugador
  - Explosión y muerte del jugador al contacto

Conexiones:

- Se conecta con D3 (Muerte del jugador)
- Se conecta con L2 (Audio explosión)

G. Enemigos - Robots y Drones (PRIORIDAD ALTA)

G1. Robot básico (punto A a B)

- [ ] G1.1 - Patrullaje punto a punto
  - Lista pública: `List<GameObject> puntosPatrulla`
  - Variable pública: `velocidadRobot`
  - Espera estática en cada punto
  - Variable pública: `tiempoEsperaEnPunto`
- [ ] G1.2 - Sistema de estados (Patrullando, Alerta, Persiguiendo)
  - Enum de estados
  - Transiciones entre estados
- [ ] G1.3 - Cono de visión con FOV
  - Variables públicas: `anguloVision`, `rangoVision`
  - Raycast para detectar jugador
  - Visualización con Gizmos
  - Detección bloqueada por obstáculos
- [ ] G1.4 - Sistema de audición
  - Detectar al jugador cuando corre
  - Variables públicas: `rangoAudicion`, `umbralRuidoDeteccion`
- [ ] G1.5 - IA de persecución
  - NavMeshAgent para seguir al jugador
  - Perder de vista y volver a patrulla
  - Variable pública: `tiempoRecuerdoPosicion`
- [ ] G1.6 - Sistema de ataque/daño
  - Atacar cuando está cerca del jugador
  - Variable pública: `rangoAtaque`, `danioAtaque`, `cooldownAtaque`

G2. Dron circular

- [ ] G2 - Igual que G1 pero con patrullaje circular/complejo
  - Lista pública: `List<GameObject> puntosPatrulla`
  - Variable pública: `velocidadDron`
  - SIN tiempo de espera en puntos (movimiento continuo)
  - Incluir G1.2 a G1.6

G3. Dron bola (más rápido)

- [ ] G3 - Similar a G1 pero más rápido
  - Lista pública: `List<GameObject> puntosPatrulla` (punto A a B)
  - Variable pública: `velocidadDronBola` (mayor que robot)
  - Incluir G1.2 a G1.6

G4. Enemigos fusionados

- [ ] G4 - Similar a G1 (punto A a B)
  - Lista pública: `List<GameObject> puntosPatrulla`
  - Variable pública: `velocidadFusionado`
  - Incluir G1.2 a G1.6
  - Posible: características especiales únicas

G5. Sistema compartido para TODOS los enemigos

- [ ] G5.1 - Script base "EnemyBase" con estados compartidos
  - Estados: Patrullando, Alerta, Persiguiendo, Stunned, Controlado
- [ ] G5.2 - Integración con habilidad de Stun (Onda Electromagnética)
  - Estado Stunned temporal
  - Variables públicas: `resistenciaStun`
- [ ] G5.3 - Integración con Control de Drones
  - Cambio temporal de facción
  - Atacar a otros enemigos cercanos
  - Restaurar comportamiento normal después

Conexiones:

- Se conecta con C1-C3 (Habilidades del jugador)
- Se conecta con D1 (Daño al jugador)
- Se conecta con I3 (Enemy Manager)
- Se conecta con L2-L3 (Audio Manager)

H. NPCs (PRIORIDAD BAJA)

- [ ] H1 - NPC estático interactuable
  - Trigger de interacción
  - Activar screamer al interactuar
- [ ] H2 - Sistema de screamers
  - Reproducir animación/sonido de susto
  - Llamar shake de cámara (B2)
  - Variable pública: `intensidadScreamer`

Conexiones:

- Se conecta con B2 (Shake cámara)
- Se conecta con E2 (Sistema interacción)
- Se conecta con L2 (Audio screamer)

I. La Jefa (PRIORIDAD BAJA - FINAL DEL JUEGO)

- [ ] I1 - Encuentro con la jefa
  - Escena/diálogo especial
- [ ] I2 - Sistema de decisión moral
  - UI con dos opciones: Desconectar / Dejar vivir
  - Afectar el final del juego
  - Variables: `jefaDesconectada` (bool)

Conexiones:

- Se conecta con J4 (UIManager para mostrar decisión)
- Se conecta con L5 (Level Manager - endings)

💻 UI / UX / HUD / MANAGERS

J. Sistema de UI Modular (PRIORIDAD MEDIA-ALTA)

J1. UIManager (Script Maestro - Singleton)

- [ ] J1.1 - Control centralizado de menús
  - Función: `AbrirMenu(TipoMenu menu)`
  - Función: `CerrarTodosMenus()`
  - Enum: `TipoMenu { Inventario, Habilidades, Pausa, Configuracion, Inspeccion }`
- [ ] J1.2 - Control de pausa del juego
  - Función: `PausarJuego(bool pausar)`
  - Modificar Time.timeScale
- [ ] J1.3 - Transiciones entre menús
  - Fade in/out
  - Animaciones de apertura/cierre
- [ ] J1.4 - Zoom de cámara al abrir menús (estilo Dead Space)
  - Llamar a sistema de cámara para zoom al hombro
  - Variables públicas: `distanciaZoomMenu`, `anguloZoomMenu`, `velocidadTransicion`

J2. InventoryUI (Script Especializado)

- [ ] J2.1 - Interfaz de inventario con secciones múltiples
  - Sección: Items importantes/consumibles
  - Sección: Información (documentos, tarjetas de acceso)
  - Navegación entre secciones con tabs
- [ ] J2.2 - Sistema de almacenamiento
  - ScriptableObjects o Dictionary para items
  - Función: `AgregarItem(Item item)`
  - Función: `EliminarItem(Item item)`
  - Función: `TieneItem(string idItem)` → bool
- [ ] J2.3 - Visualización de items en grid/lista
  - Prefabs de UI para cada tipo de item
  - Iconos, nombres, descripciones
- [ ] J2.4 - Inspector 3D de modelos (tipo Valorant)
  - Función: `InspeccionarModelo3D(GameObject modelo)`
  - Cámara secundaria para renderizar modelo
  - Rotación del modelo con mouse drag
  - Zoom in/out con scroll
  - Visualización de detalles/texturas/materiales
  - Variables públicas: `velocidadRotacion`, `zoomMin`, `zoomMax`

J3. HabilidadesUI (Script Especializado)

- [ ] J3.1 - Selector de habilidades estilo Dead Space
  - Mostrar 3 habilidades disponibles
  - Indicador visual de habilidad seleccionada actualmente
  - Selección con mouse o teclas numéricas
- [ ] J3.2 - Visualización de cooldowns en tiempo real
  - 3 barras circulares o radiales
  - Sincronización con cooldowns de C1-C3
  - Función: `ActualizarCooldown(int idHabilidad, float progreso)`
  - Animación de recarga $(0 \rightarrow 1)$
- [ ] J3.3 - Estilo holográfico Dead Space
  - Shader holográfico/transparente
  - Efectos de brillo/scanlines
  - Animación de apertura/cierre

J4. InteractionUI (Script Especializado)

- [ ] J4.1 - Prompt de interacción
  - Texto dinámico: "Presiona [E] para interactuar"
  - Aparecer/desaparecer según proximidad
  - Función: `MostrarPrompt(string texto, bool mostrar)`
- [ ] J4.2 - Estilo visual
  - Fade in/out suave
  - Posición en pantalla configurable
  - Variable pública: `tiempoFade`

J5. CinematicUI (Script Especializado)

- [ ] J5.1 - Reproducción de cinemáticas 2D (videos)
  - Integración con Video Player de Unity
  - Trigger al recoger objetos especiales de información
  - Función: `ReproducirCinematica(VideoClip video)`
- [ ] J5.2 - Control durante reproducción
  - Pausar gameplay completamente
  - Barras negras cinematográficas (letterbox)
  - Botón de skip (opcional)
  - Reanudar gameplay al terminar
- [ ] J5.3 - Registro de videos vistos
  - Lista: `videosRecolectados`
  - Poder revisar videos desde inventario

Conexiones:

- J1 se conecta con TODOS los demás scripts de UI (maestro)
- J2 se conecta con B5, E3 (Cámara zoom, recolección)
- J3 se conecta con C1-C4 (Habilidades)
- J4 se conecta con E1 (Interacción)
- J5 se conecta con E3 (Objetos especiales)
- Todos se conectan con L1 (Game Manager)

K. HUD (PRIORIDAD MEDIA)

- [ ] K1 - Indicador de vida
  - Manejado por N1 (PostProcessManager) con vignette
  - Opcional: barra de vida minimalista adicional
- [ ] K2 - Indicador de estado del jugador
  - Iconos simples: agachado, escondido, detectado
  - Variables públicas: `mostrarIndicadores` (bool)
- [ ] K3 - Crosshair/Mira
  - Aparecer solo en modo apuntar
  - Estilo configurable
  - Variable pública: `tipoCrosshair`

Conexiones:

- Se conecta con A2-A3 (Estados del jugador)
- Se conecta con B5 (Apuntar)
- Se conecta con N1 (PostProcessManager)

🎨 POST-PROCESSING (PRIORIDAD ALTA - NUEVO)

N. PostProcessManager (Script Único - Singleton)

- [ ] N1.1 - Configuración inicial
  - Referencia a Global Volume en escena
  - Obtener componentes: Vignette, ColorGrading, ChromaticAberration, Blur
  - Singleton pattern para acceso global
- [ ] N1.2 - Sistema de vignette de daño
  - Función: `SetDamageVignette(float vidaPorcentaje)`
  - Intensidad basada en porcentaje de vida (0-1)
  - Variables públicas: `intensidadVignetteMin`, `intensidadVignetteMax`
  - Transición suave con Lerp
  - Color: rojo para daño
- [ ] N1.3 - Vignette de muerte
  - Función: `ActivarVignetteMuerte()`
  - Vignette al 100% cubriendo toda la pantalla
  - Color: rojo oscuro/negro
- [ ] N1.4 - Restauración gradual de vignette (QTE exitoso)
  - Función: `LerpVignetteNormal(float velocidad)`
  - Coroutine para transición suave a estado normal
  - Variable pública: `velocidadRestauracion`
- [ ] N1.5 - Blur de menús
  - Función: `ActivarBlurMenu(bool activo)`
  - Blur de fondo cuando se abren menús (inventario, habilidades)
  - Variable pública: `intensidadBlur`
- [ ] N1.6 - Efectos adicionales opcionales
  - Chromatic aberration para stun/aturdimiento
  - Color grading para ambientes (frío, cálido)
  - Función: `AplicarEfectoStun(float duracion)`
- [ ] N1.7 - Sistema de reseteo
  - Función: `ResetearEfectos()`
  - Volver todos los efectos a valores por defecto
  - Útil al cargar checkpoint o reiniciar

⚠️ IMPORTANTE:

- Este script maneja TODO el post-processing del juego
- Solo debe haber UNA instancia (Singleton)
- Otros scripts solo LLAMAN funciones, NO modifican directamente el Volume
- Usar UnityEngine.Rendering.Universal (URP) o HighDefinition (HDRP)

Conexiones:

- Se conecta con D1-D4 (Sistema de vida y daño)
- Se conecta con J1 (UIManager - blur de menús)
- Se conecta con C1-C3 (Efectos de habilidades opcionales)
- Se conecta con L1 (Game Manager - reseteo)

🔧 MANAGERS GENERALES (PRIORIDAD ALTA - SISTEMA CENTRAL)

L1. Game Manager (Singleton)

- [ ] L1.1 - Control de estado del juego
  - Estados: Menu, Jugando, Pausado, GameOver
  - Singleton pattern
- [ ] L1.2 - Sistema de checkpoints
  - Guardar posición del jugador en checkpoint
  - Guardar inventario y estado del nivel en checkpoint
  - Función: `GuardarCheckpoint()`, `CargarCheckpoint()`
  - Limpiar inventario y progreso al morir y respawnear
  - Llamar a `PostProcessManager.ResetearEfectos()` al cargar
- [ ] L1.3 - Sistema de guardado persistente (opcional)
  - Guardar progreso entre sesiones (PlayerPrefs o JSON)

L2. Audio Manager (Singleton)

- [ ] L2.1 - Control de música de fondo
  - Variables públicas: `volumenMusica`
  - Transiciones entre tracks
  - Función: `CambiarMusica(AudioClip clip, float fadeTime)`
- [ ] L2.2 - Control de efectos de sonido
  - Variables públicas: `volumenEfectos`
  - Sistema de AudioClips en pool para optimización
  - Función: `ReproducirSFX(AudioClip clip, Vector3 posicion)`
- [ ] L2.3 - Control de sonidos ambientales/diálogos
  - Variables públicas: `volumenAmbiente`
- [ ] L2.4 - Integración con menú de configuración
  - Sliders para cada tipo de audio
  - Guardar preferencias con PlayerPrefs

L3. Animation Manager / Controller

- [ ] L3.1 - Sistema de animaciones del jugador
  - Animator Controller con estados: Idle, Walk, Run, Crouch, etc.
  - Blend trees para movimiento suave
  - Función: `SetAnimationState(string estado)`
- [ ] L3.2 - Sistema de animaciones de enemigos
  - Estados: Patrol, Alert, Chase, Attack, Stunned
  - IK para apuntar al jugador (opcional)

L4. Enemy Manager (Singleton)

- [ ] L4.1 - Registro de todos los enemigos en escena
  - Lista de enemigos activos: `List<EnemyBase> enemigosActivos`
  - Función: `RegistrarEnemigo(EnemyBase enemigo)`
  - Función: `DesregistrarEnemigo(EnemyBase enemigo)`
- [ ] L4.2 - Sistema de alerta global (opcional)
  - Cuando un enemigo detecta al jugador, alertar a cercanos
  - Variable pública: `rangoAlertaGlobal`
  - Función: `AlertarEnemigosEnRango(Vector3 posicion)`

L5. Level Manager

- [ ] L5.1 - Control de carga de niveles/escenas
  - Transiciones entre niveles
  - Loading screens con barra de progreso
  - Función: `CargarNivel(string nombreNivel)`
- [ ] L5.2 - Sistema de puertas con llaves
  - Verificar tarjetas de acceso en inventario
  - Abrir/cerrar puertas con animación
  - Función: `AbrirPuerta(string idPuerta, string idLlave)`
- [ ] L5.3 - Activación/desactivación de objetos del nivel
  - Resetear objetos al cargar checkpoint
  - Sistema de triggers y eventos del nivel

Conexiones GLOBALES:

- L1 se conecta con TODOS los sistemas (GameManager central)
- L2 se conecta con M1 (Configuración audio)
- L3 se conecta con A1-A3, G1-G4 (Animaciones)
- L4 se conecta con G1-G4 (Enemigos)
- L5 se conecta con E3, J2 (Llaves e inventario)

M. Sistema de Configuración (PRIORIDAD BAJA)

- [ ] M1 - Menú de opciones
  - Sliders de volumen (música, efectos, ambiente)
  - Sliders de sensibilidad de cámara (X, Y)
  - Toggle de efectos gráficos (opcional)
  - Guardar configuración con PlayerPrefs
  - Función: `CargarConfiguracion()`, `GuardarConfiguracion()`
- [ ] M2 - Aplicar configuración en tiempo real
  - Conectar con Audio Manager (L2)
  - Conectar con sistema de cámara (B1)
  - Sin necesidad de reiniciar

Conexiones:

- Se conecta con B1 (Sensibilidad cámara)
- Se conecta con L2 (Volúmenes audio)
- Se conecta con J1 (UIManager para mostrar menú)

📝 NOTAS IMPORTANTES PARA EL EQUIPO

Arquitectura de Scripts UI/Post-Processing:

```
PostProcessManager.cs (Singleton) ← UN SOLO SCRIPT para TODO el post-processing
 ├── Vignette de daño
 ├── Vignette de muerte
 ├── Blur de menús
 └── Efectos adicionales

UIManager.cs (Singleton - Maestro) ← Controla apertura/cierre de menús
 ├── InventoryUI.cs ← Maneja solo el inventario
 ├── HabilidadesUI.cs ← Maneja solo las habilidades
 ├── InteractionUI.cs ← Maneja solo los prompts
 └── CinematicUI.cs ← Maneja solo los videos
```

Prioridades Sugeridas (Si hay prisa):

1. FASE 1 (MVP Jugable): A1-A3, B1, D1-D3, N1.1-N1.3, G1 (solo patrullaje básico), L1, L2, J1
2. FASE 2 (Gameplay Core): B2-B7, C1-C4, G1-G4 (completos con IA), E1-E3, N1.4-N1.7, J2-J5, K1-K3
3. FASE 3 (Polish y Features): E4, H1-H2, I1-I2, M1-M2, L3-L5

Sistema de Nomenclatura:

- A-E: Personaje jugable
- F-I: Enemigos y NPCs
- J-K: UI/HUD
- N: Post-Processing (NUEVO)
- L-M: Managers

Dependencias Críticas:

- L1 (Game Manager) debe existir ANTES de empezar cualquier otro sistema
- N1 (PostProcessManager) debe existir ANTES de implementar sistema de vida
- J1 (UIManager) debe existir ANTES de los scripts especializados de UI
- A1 (Movimiento) y B1 (Cámara) son la base de todo
- G5 (Sistema base enemigos) debe hacerse antes de los enemigos individuales

Tips de Optimización:

- Usar Object Pooling para enemigos y proyectiles
- NavMesh para pathfinding de enemigos
- Cinemachine para cámaras dinámicas
- Post-Processing Stack V2 (URP) o HDRP
- Event System o UnityEvents para desacoplar sistemas
- Singleton pattern para Managers (evitar FindObjectOfType)

Ejemplo de Llamadas entre Scripts:

```csharp
// PlayerHealth.cs llama a PostProcessManager
PostProcessManager.Instance.SetDamageVignette(vidaActual / vidaMaxima);

// UIManager.cs llama a PostProcessManager
PostProcessManager.Instance.ActivarBlurMenu(true);

// PlayerInteraction.cs llama a InventoryUI
InventoryUI.Instance.AgregarItem(itemRecolectado);
```

✅ CHECKLIST DE INTEGRACIÓN

Antes de dar por terminada una tarea, verificar:

- [ ] ¿Tiene todas las variables públicas mencionadas?
- [ ] ¿Tiene Gizmos si fue requerido?
- [ ] ¿Está comentado el código?
- [ ] ¿Se probó en Play Mode?
- [ ] ¿Usa Singleton correctamente si es Manager?
- [ ] ¿Se comunicó con el equipo sobre las conexiones con otros scripts?
- [ ] ¿Los scripts especializados NO modifican directamente lo que maneja otro Manager?

¡Suerte con el proyecto! 🎮🚀