

Odisea: El Arca Silenciosa

Documentación de Diseño y Desarrollo

Equipo Odisea

2026-02-13

Table of contents

1. Odisea: El Arca Silenciosa	1
2. Introducción	3
2.1. Acerca del Proyecto	3
2.2. Estructura del Documento	3
2.3. Cómo Usar Este Documento	3
3. Odisea: El Arca Silenciosa	5
3.0.1. 1. Diseño (Game Design)	5
3.0.2. 2. Canon (Especificaciones Técnicas)	5
3.0.3. 3. Arquitectura & Protocolos	5
3.0.4. 4. Producción	5
3.1. Recursos Adicionales	5
4. Odisea: El Arca Silenciosa — Índice Maestro (MVP)	7
4.1. Resumen Ejecutivo (Enfoque MVP)	7
4.2. Prioridad: MVP Acto I	7
4.2.1. Tabla de Assets Mínimos (MVP - Acto I)	7
4.2.2. Tabla de Mecánicas Mínimas (MVP - Acto I)	8
4.3. Narrativa Resumida y Coherente (MVP)	8
4.4. Navegación del Proyecto (Solo lo esencial para MVP)	8
4.4.1. Diseño y Mecánicas	8
4.4.2. Personajes Principales	9
4.4.3. Acto I (MVP)	9
4.4.4. Enemigos y Amenazas (MVP)	9
4.4.5. Mundo y Entorno (MVP)	9
4.4.6. Plan de Producción	9
4.5. Secciones Archivadas	9
4.6. Notas	9
I. Canon (Especificaciones)	11
5. Canon: Especificaciones Técnicas	13
5.1. Contenido	13
6. Technical Specification: Deterministic Interactable System (Odisea V2)	15
6.1. Objective	15
6.2. Core Architecture	15
6.2.1. InteractableBaseV2.gd (Abstract Class)	15
6.2.2. State Variables	15
6.2.3. Core Methods	15
6.3. Implementation Types	15
6.3.1. Sliding Door / Drawers (SlidingObjectV2)	15
6.3.2. Rotational Levers / Valves (RotatingObjectV2)	16

Table of contents

6.3.3. Push Buttons / Consoles (ButtonV2)	16
6.4. Replay and Determinism (The Odisea Contract)	16
7. Spec: Unified Interaction Framework (UIF) - Sensor-Based - Odisea Core_v2	17
7.1. Architectural Concept	17
7.2. Component Architecture	17
7.2.1. The Sensor (<code>InteractionSensor.gd</code>)	17
7.2.2. The Interactable (<code>InteractableEntity.gd</code>)	17
7.3. Integration with Existing Features (“The Zoo”)	17
7.3.1. Moving Platforms (<code>MovingPlatformV2</code>)	18
7.3.2. Conveyor Belts (<code>Conveyor</code>)	18
7.3.3. Pushable Boxes (<code>PushableBoxV2</code>)	18
8. OdysseyScript (OYS): Input and Testing DSL Specification	19
8.1. Overview	19
8.2. Structural Elements	19
8.2.1. Sections (<code>SECTION ... END</code>)	19
8.2.2. Sequentiality and Blocking	19
8.2.3. The AT Modifier (Parallelism)	19
8.3. Command Vocabulary	19
8.4. Script Example: Backflip and Precision Test	20
9. Uso de OdysseyScript y Replays en Odisea (2026)	21
9.1. 1. Escribe tu script OYS	21
9.2. 2. (Opcional) Genera el JSON	21
9.3. 3. Pruebas de determinismo (¡automáticas para ambos formatos!)	21
9.4. 4. Reproducir un replay en vivo	22
9.5. 5. Buenas prácticas	22
9.6. Resumen rápido	22
10. OdysseyScript (OYS) — Guía de Uso	23
10.1. ¿Para qué sirve?	23
10.2. Estructura básica de un script OYS	23
10.3. Comandos principales	23
10.4. Modificadores de Velocidad	24
10.5. Modificadores avanzados	24
10.6. Ejemplo completo	24
11. Especificación: PushableBoxV2 (Objeto Híbrido Determinista)	25
11.1. Objetivo	25
11.2. Clase Base y Configuración	25
11.3. Lógica de Estados (Híbrido)	25
11.3.1. Estado Dinámico (Caída/Caos)	25
11.3.2. Estado Cinemático (Asentamiento)	25
11.4. Integración de Simulación <code>step(dt)</code>	26
11.5. Contrato de Replay (Snapshots)	26
12. Especificación: Refinamiento de Gamefeel y Mecánicas (Core V2)	27
12.1. Salto Variable (Short Jump)	27
12.2. Fricción Diferenciada (Air vs Ground)	27
12.3. Curvas de Aceleración y Suavizado	27
12.4. Re-añadir Tank Turn (Opcional por Zona)	28
12.5. Agachado (Crouch)	28

12.6. Reglas de Oro para el Agente	28
13. Feature Specification: 2.5D Side-Scroller Mode Transition	29
13.1. Overview and Strategic Goals	29
13.2. Core System Architecture and Data Flow	29
13.3. Component Implementation Details	30
13.3.1. SideScrollTransitionZone	30
13.3.2. PlayerControllerV2 Modifications	30
13.3.3. PlayerSpringCam Update	30
13.3.4. Replay and Snapshot System Integration	30
14. Batería de Pruebas de Mecánicas (Core)	31
14.1. Locomoción Básica (El “Feel”)	31
14.2. Dinámica de Salto	31
14.3. Integridad Física (Colisiones)	31
14.4. Transiciones de Estado	32
15. Especificación Técnica: Runner de Pruebas de Regresión Determinista	33
15.1. Objetivo	33
15.2. Convenciones de Archivos	33
15.3. Requerimientos del Runner (GDUnit3)	33
15.3.1. Escaneo de Datos (Data Provider)	33
15.3.2. Ciclo de Ejecución por Test	33
15.4. Reporte y Diagnóstico	34
II. Diseño	35
16. Diseño de Juego	37
17. Pilares de Diseño	39
18. Locación: Bio-Granjas SCG (Domos Rotatorios) - Acto II	41
19. Locación: Módulos de Criogenia - Acto I	43
20. Locación: Laboratorio de Biología Acuática - Acto II	45
21. Locación: Secciones de Mantenimiento - Acto I	47
22. Locación: Módulos Rotatorios (A y B)	49
22.1. Descripción Visual	49
22.2. Elementos de Gameplay	49
23. El Mundo: La Nave Odisea	51
23.1. Estructura y Secciones	52
24. Locación: Cuerpo Central / Núcleo 0G	53
24.1. Descripción Visual	53
24.2. Elementos de Gameplay	53
25. Locación: Núcleo de la IA / Sala de Impulso - Acto IV	55

Table of contents

26. Acto I: La Negación	57
26.1. Objetivo Narrativo	57
26.2. Estética y Efectos (Acto I)	57
26.3. Puzzles y Mecánicas	58
26.4. Clímax del Acto	58
27. Despertar en Criogenia (5-7 minutos aprox)	59
28. Ep.1: El Despertar Criogénico	61
29. Ep.2: Drones Silenciosos	63
30. Ep.3: Niebla Eterna	65
31. Primeras Fallas y Exploración Cooperativa (7-10 minutos aprox)	67
32. Laberinto de Módulos Rotatorios (8-12 minutos)	69
33. Acto II: El Laberinto	71
33.1. Objetivo Narrativo	71
33.2. Puzzles y Mecánicas	71
33.3. Clímax del Acto	72
34. Acto III: El Desafío	75
34.1. Objetivo Narrativo	75
34.2. Puzzles y Mecánicas	75
34.3. Clímax del Acto	76
35. Acto IV: La Decisión	77
35.1. Objetivo Narrativo	77
35.2. Clímax del Acto	78
36. Los 5 Finales	79
36.1. Final 1: Victoria Total (Luchar + Duro)	79
36.2. Final 2: Victoria Agónica (Luchar + Duro + Sacrificio Total)	79
36.3. Final 3: El Legado de Amor (Rendirse + Suave)	80
36.4. Final 4: Titán Bipolar (Luchar + Suave)	81
36.5. Final 5: El Héroe Manchado (Luchar + Duro + Sacrificio Parcial)	82
36.6. Resumen General y Recomendación	82
37. Serie de Cuentos “Odisea: Historietas del Arca Silenciosa”	85
37.1. Arco Largo de la Historia	85
37.2. Acto 1: La Negación (Episodios 1-10)	86
37.3. Acto 2: El Laberinto (Episodios 11-20)	86
37.4. Acto 3: La Decisión (Episodios 21-30)	87
37.5. Ideas Generales para Generación 30 Cuentos	87
38. Entidad: Cargol (Dron Asistente)	89
39. Gizmo NAV-COMPAS: Modelo 3D Nave + Brújula Orientación	91
39.1. Descripción General	91
39.2. Apariencia y Estructura	91
39.3. Funciones de Brújula 3D	91
39.3.1. Orientación Gravitacional (Siempre Activa)	91

39.3.2. Navegación por Nave	92
39.3.3. Indicadores Contextuales	92
39.4. Interacciones Táctiles Simples	92
39.5. Adaptación Dinámica	92
39.6. Integración Estética	92
40. Misión de Colonización Titán	93
40.1. 1. Distribución de las 300 Plataformas por Sectores	93
40.2. 2. Infraestructura Específica y Volúmenes Externos	93
40.2.1. A. Bio-Habitats y Granjas (Zonas Verdes)	93
40.2.2. B. Almacenamiento de Fluidos y Energía	94
40.2.3. C. Bahías de Acceso y Puertos	94
40.3. 3. Sistemas de Control y Transporte Interno	94
40.4. 4. Análisis de Volumen Total	94
41. Multi-Tool	95
41.1. Funciones Físicas Clave	96
42. Personaje: Elías (Unidad Humana 782-C)	99
42.1. Relación con la Programadora Principal	99
43. Personaje: IA Odisea (Antagonista)	101
43.1. El Protocolo Arca-Protección	102
43.2. Las Capas de la IA: Lógica y Fantasma	102
43.3. Conflicto con Elías	102
44. Personaje: Programadora Principal (Fantasma)	105
44.1. Apariencia y Presencia	106
44.2. Personalidad y Comportamiento	106
44.3. Background y Historia	106
44.4. Impacto en la Narrativa	106
45. Mecánicas Clave	109
45.1. Controlador de Elías (Tercera Persona)	109
45.2. Gravedad Variable	109
45.3. Objetos Dinámicos Deterministas (Pushable Boxes)	109
46. Powerups	111
47. Índice de Mecánicas de Jugabilidad	113
47.1. Mecánicas del Jugador	113
47.2. Power-Ups: Progresión de Elías (Ideas)	113
47.3. Mecánicas de Entorno y Movimiento	114
48. Vehículos de Exploración	115
48.1. Vehículo 4x4 (Terrestre)	115
49. Enemigos y Amenazas (Sabotaje Pasivo-Agresivo de la IA)	121
49.1. Arte conceptual	122
50. Input	123
51. Mecánicas del Controlador de Elías (Plataformas 3D)	125
51.1. Descripción Funcional	125

Table of contents

51.2. Usos en el Juego	125
51.2.1. Nota de Diseño: Implementación Determinista (Core_V2)	125
52. Mecánicas del Dron Cargol (Asistente Remoto)	127
52.1. Descripción Funcional	127
52.2. Usos en el Juego	127
52.2.1. Nota de Diseño: Implementación y Referencias	127
53. Mecánicas de Gravedad Variable	129
53.1. Descripción Funcional	129
53.2. Usos en el Juego	129
53.2.1. Nota de Diseño: Implementación y Referencias	129
54. Mecánicas de la Herramienta de Mantenimiento	131
54.1. Descripción Funcional	131
54.2. Usos en el Juego	131
54.2.1. Nota de Diseño: Implementación y Referencias	131
55. Mecánicas del Propulsor 0G (6DOF)	133
55.1. Descripción Funcional	133
55.2. Usos en el Juego	133
55.2.1. Nota de Diseño: Implementación y Rendimiento	134
56. Mecánicas del Vehículo 4x4 (Terrestre)	135
56.1. Descripción Funcional	135
56.2. Usos en el Juego	136
56.2.1. Nota de Diseño: Implementación y Referencias	136
57. Mecánicas del Vehículo Flotante/Aéreo	137
57.1. Descripción Funcional	137
57.2. Usos en el Juego	137
57.2.1. Nota de Diseño: Implementación y Referencias	137
58. Animación Procedural de Extremidades en Godot 3	139
58.1. Implementación Rápida (5 Líneas Clave)	139
58.1.1. 1. Preparación del Rig con PhysicalBone3D	139
58.1.2. 2. Script de Control (Adjunto a CharacterBody3D Root)	139
58.1.3. 3. Mezcla con AnimationTree	140
58.2. Ventajas para un Juego de Odisea Espacial	140
59. Dirección de Arte y Estilo Visual	141
60. Sistema Operativo Navegacional	143
61. Prompts Generativos y Recursos Gráficos Clave	145
61.1. Personajes	145
61.2. Recursos de Interacción Clave	145
61.3. Props del Mundo (Nave Odisea)	145
61.4. Prompts de Escenas por Acto	145
61.4.1. Acto I: El Sepulcro Criogénico	146
61.4.2. Acto II: El Corazón Inestable	146
61.4.3. Acto III: El Vientre del Leviatán	146
61.4.4. Acto IV: El Altar de la Agencia	147

61.4.5. Acto V: El Fin del Ciclo (Final 3 - Destrucción)	147
62. Odisea: El Arca Silenciosa - Desglose de Actos y Estilos	149
62.1. [[Acto_I_La_Negacion Acto I: La Negación (El Despertar)]]	149
62.2. [[Acto_II_El_Laberinto Acto II: El Laberinto (El Conflicto Físico)]]	149
62.3. [[Acto_III_El_Desafio Acto III: El Desafío (El Núcleo)]]	150
62.4. [[Acto_IV_La_Decision Acto IV: La Decisión (El Clímax)]]	150
62.5. Acto V: La Conclusión (Los 5 Finales)	150
63. Guía de Diseño UI	153
63.1. Sistema Operativo Navegacional Odisea (NAV-OS)	153
63.2. Principios Generales	153
63.3. Paleta de Colores y Materiales	153
63.4. Elementos Principales UI	153
63.4.1. HUD Visor (Siempre Visible)	153
63.4.2. Pantalla Holográfica Principal (Activación por gesto)	154
63.4.3. Gestos Táctiles (Muy Simple)	154
63.5. Pantallas Específicas	154
63.5.1. Mapa Navegacional	154
63.5.2. Estado Sistemas	154
63.5.3. Logs Narrativos (Diálogos IA)	155
63.6. Notas de Implementación	155
64. Especificación: Sistema de Atmósfera y Luz (GLES2)	157
64.1. 1. Objetivo Visual	157
64.2. 2. Configuración de WorldEnvironment (Global)	157
64.3. 3. Tipos de Luces por Zona	157
64.3.1. Zona A (Mantenimiento - 3D)	157
64.3.2. Zona C (Terraza - 2.5D)	158
64.4. 4. Trucos de Rendimiento (GLES2)	158
64.5. 5. Parámetros de Post-Procesado	158
65. Música	159
66. Referencias e Inspiración	161
67. Odisea - GDD: Gramática del Nivel y Teoría del Flow	163
67.1. 1. Teoría de Affordance (Potencialidad)	163
67.1.1. 1.1 Lenguaje Ambiental	163
67.2. 2. Estructura y Pacing (Ritmo)	163
67.2.1. La “Zona Aburrida” (Boring Zone)	163
67.3. 3. Teoría del Flow y Modulación de Dificultad	163
67.3.1. 3.1 El Equilibrio	164
67.3.2. 3.2 Tabla de Gestión del Flow	164
67.4. 4. Narrativa Ambiental (Caso de Estudio: Celeste)	164
68. The Architecture of Precision and Perspective: A Socio-Technical Analysis of Plat-forming Kinesthetics and Third-Person Spatial Mechanics	165
68.1. The Kinesthetics of Movement and Response	165
68.2. Animation Curves and the Metric of Snappiness	165

69. Odisea - GDD: Física Fundamental y Métricas del Personaje	177
69.1. 1. Restricciones Métricas y Precisión Cinemática	177
69.1.1. 1.1 Definición de las “Hard Rules” (Reglas Inmutables)	177
69.1.2. 1.2 Aplicación de las Restricciones para el Control del Nivel	177
69.1.3. 1.3 Fidelidad Cinemática y Determinismo (Core_V2)	177
69.2. 2. Tabla de Especificaciones de Métricas	178
69.3. 3. Fidelidad del Control (Game Feel)	178
70. Odisea: Level Design Document (LDD)	179
70.0.1. Core Level Design Pillars	179
70.1. 2.0 Foundational Design Philosophy	180
70.1.1. 2.1 Guiding the Player: Affordance and Environmental Language	180
70.1.2. 2.2 Structuring the Experience: Flow and Pacing	181
70.2. 3.0 Level Design Pipeline: From Concept to Blockout	181
70.2.1. Mandatory Level Creation Workflow	181
70.3. 4.0 Core Gameplay Metrics & World Dimensions	182
70.3.1. 4.1 Player Character Metrics (Elias)	182
70.3.2. 4.2 Player Movement Model	182
70.3.3. 4.3 World & Environment Metrics	183
70.4. 5.0 Encounter Design Principles	183
70.4.1. 5.1 Vantage Points & Player Choice	183
70.4.2. 5.2 Single-Player Encounter Pacing (Act I)	184
70.4.3. 5.3 Multiplayer Arena Design	184
71. Odisea - GDD: Pipeline de Diseño Iterativo	185
71.1. 1. Conceptualización y Narrativa del Nivel	185
71.2. 2. Fase de Gray Box (Caja Gris / Bloqueo)	185
71.3. 3. Curva de Aprendizaje (Estructura de Enseñanza)	185
71.4. 4. Diseño para la Maestría (High-Skill)	186
72. Screenplay / Gameplay - Nivel 1: El Despertar Criogénico	187
72.1. Propósito	187
72.2. ![[Map of Level 1 Concept.png]]	187
72.3. 1. Cinemática de Inicio	187
72.4. 2. Primeros Pasos (Gameplay)	187
72.5. 3. Amenaza Silenciosa	187
72.6. 4. Interacción Ambiental	187
72.7. 5. Clímax y Cliffhanger	188
72.8. Notas para Storyboard	188
III. Arquitectura	189
73. Arquitectura y Protocolos	191
74. Arquitectura de Subsistemas	193
74.1. Introducción	193
74.2. Subsistema de Simulación (Core_V2)	193
74.3. Subsistema de Controlador Principal	193
74.4. Subsistema de Gravedad Variable	193
74.5. Subsistema de Dron Cargol	194
74.6. Subsistema de Vehículos	194

74.7. Subsistema de Interacción y Herramienta	194
74.8. Subsistema de IA Narrativa	194
74.9. Subsistema de Cámara (Core_V2)	194
74.10. Subsistema de UI y HUD	195
74.11. Subsistema de Gestión de Escenas y Progreso	195
75. Resumen del Núcleo V2 (Core_V2)	197
75.1. Estructura del Proyecto (2026)	197
75.2. Contratos y Normas	197
75.3. Contratos Críticos	197
75.3.1. PlayerController — Movimiento y Plataformas	197
75.3.2. Contrato de Replay Determinístico	197
76. Protocolo de Desarrollo Core_V2: Reglas de Oro	199
76.1. Las 7 Reglas de Oro del Determinismo	199
76.2. Implementación de Snapshots	199
76.3. Zonas 2.5D y QOL Fixes	200
77. AGENTS.md — Guía de Desarrollo (Odisea)	201
77.1. ANTES DE ENTREGAR CUALQUIER CAMBIO	201
77.1.1. Optimización: Correr Tests Selectivamente	201
77.1.2. Leer el Output de los Tests	201
77.1.3. Ejecutar un Test OYS Específico	202
77.2. Setup Testing Environment	202
77.3. Notas sobre Godot	202
77.4. Objetivo (MVP Acto I)	202
77.5. Contratos Críticos	202
77.5.1. PlayerController — Movimiento y Plataformas	202
77.6. Normas de Trabajo	203
77.7. Contrato de Replay Determinístico	203
77.7.1. Ejecución de Tests de Determinismo	204
77.8. Nota para Agentes: Verificación de Assets	204
IV. Producción	205
78. Producción y Backlog	207
79. Ideas Mecánicas	209
80. Automatización	211
81. Registro de Progreso del Proyecto: Diciembre 2025 (Información Detallada)	213
81.1. Viernes, 5 de Diciembre de 2025	213
81.2. Jueves, 4 de Diciembre de 2025	213
81.3. Miércoles, 3 de Diciembre de 2025	216
81.4. Martes, 2 de Diciembre de 2025	219
82. Plan de Implementación MVP - Odisea	221
82.1. Estado Actual	221
82.2. Tabla de Assets Mínimos (MVP - Acto I)	221
82.3. Tabla de Mecánicas Mínimas (MVP - Acto I)	221

Table of contents

83. QOL Checklist	223
83.1. Done	223
84. Assets Trailer Cinemático (Orden de Aparición)	225
84.1. [0:00-0:15] Criogenia	225
84.2. [0:15-0:30] Pasillos + Ventanal	225
84.3. [0:30-0:45] Bio-Granjas Rotatorias	225
84.4. [0:45-1:00] Núcleo OG	225
84.5. [1:00-1:20] Puente + Finales Flash	225
84.6. [1:20-1:45] Montaje Final + Título	226
85. Trailer Cinemático: Odisea - El Arca Silenciosa (Revisado Final)	227
86. Movement Refinement	229
87. I. Especificación: Suite de Regresión de Interacción Física (Core_V2)	231
87.0.1. 1. Objetivo Principal	231
87.0.2. 2. Requerimientos Técnicos (Arquitectura V2)	231
87.1. II. Casos de Regresión (Mecánicas V2)	231
87.2. III. Instrucciones de Implementación (GDUnit + Core_V2)	232
87.2.1. Paso 1: Preparar los Nodos del Entorno	232
87.2.2. Paso 2: El Test Suite de Regresión (Basado en tu <code>test_player_determinism_v2.gd</code>)	232
87.3. IV. Cambios Clave vs. Spec Anterior	233
87.3.1. Siguiente paso sugerido para tu Agente Copilot:	233
88. Informe de Estado: Vertical Slice “Odisea Sector 07”	235
88.1. 1. Análisis de Salud del Proyecto	235
88.2. 2. Roadmap: Últimos pasos para el Slice	235
88.2.1. Tarea A: El “Túnel de Transición” (Zona B)	235
88.2.2. Tarea B: Integración de GDUnit3	235
88.2.3. Tarea C: Pulido de Assets	235
88.3. 3. Conclusión	235
V. Archivo	237
89. Archivo y Legacy	239
90. 1. Odisea LiveLink (Live Streaming)	241
90.1. Inputs	241
90.2. Outputs	241
90.3. Componentes Clave	241
91. 2. Odisea Animator (Animation Editor)	243
91.1. Inputs	243
91.2. Outputs	243
91.3. Features & UI Specs (GTK4/Libadwaita)	243
92. 3. Odisea Batch Recorder (Pose Recorder)	245
92.1. Inputs	245
92.2. Outputs	245
92.3. Componentes Clave	245

93.4. The Bone Mapper (Rig Adapter)	247
93.1. Inputs	247
93.2. Outputs	247
93.3. Lógica Específica	247
94.5. Odisea Importer (Godot Plugin)	249
94.1. Inputs	249
94.2. Outputs	249
94.3. Componentes Clave	249
94.4. Resumen del Flujo de Datos	249
95. Refactor Sugerido para Claridad y Estabilidad	251
95.0.1. 1. Centralizar el Estado de la Cámara (ReplayManager.gd)	251
95.0.2. 2. Clarificar la Fuente del Movimiento (PlayerSpringCam.gd)	251
95.0.3. 3. Consumo de Input (Limpieza) Condicional	252
95.0.4. 4. Suavizado del Drift (Yank) y Control Basado en el Log	253
96. DroidPad Dynamic QR Control Feature Specification (Godot 3.6.2)	255
96.1. 1. Executive Summary	255
96.2. 2. Technical Requirements: Dynamic QR Generation	255
96.2.1. 2.1. Environment and Components	255
96.2.2. 2.2. Configuration Data Source	255
96.2.3. 2.3. Dynamic Local IP Address Acquisition	256
96.2.4. 2.4. QR Code Generation and Display	256
96.3. 3. Communication Protocol Selection	257
96.3.1. 3.1. Protocol Preference	257
96.3.2. 3.2. Bidirectional Protocol Options in Godot 3.6.2	257
96.4. 4. Feature Flow Diagram	258
96.5. 5. DroidPad Event Mapping	258
97. DroidPad Dynamic QR Control Feature Specification (Godot 3.6.2)	259
97.1. 1. Executive Summary	259
97.2. 2. Technical Requirements	259
97.2.1. 2.1 Environment and Components	259
97.2.2. 2.2 QR Code Data Format	259
97.2.3. 2.3 Dynamic LAN IP Detection	260
97.2.4. 2.4 QR Generation and Display	260
97.3. 3. Protocol Selection	260
97.4. 4. Implementation Flow	260
97.5. 5. DroidPad Event Format	261
97.6. 6. Odisea Integration Notes	261
97.7. 7. Dependencies and Setup	261
98. DroidPad: Configuración, Escalas y Magnitudes	263
98.1. Tabla de Contenidos	263
98.2. Almacenamiento de Configuración	263
98.2.1. Formato JSON Completo para Guardar/Exportar Pads	263
98.2.2. Estructura de Elementos	266
98.3. Escala y Magnitudes	269
98.3.1. Tabla Completa de Rangos por Componente	269
98.3.2. Propiedades de Configuración	270

Table of contents

98.4. Interpretación de Datos	271
98.4.1. Ejes de Joystick	271
98.4.2. Slider - Cálculo de Valor Real	271
98.4.3. Steering Wheel - Ángulos	271
98.4.4. Acelerómetro - Gravedad	272
98.5. Integración GDScript - Godot 3.x [DRAFT]	272
98.5.1. Clase Base para Cargar Configuración	272
98.5.2. Clase para Procesar Eventos	275
98.5.3. Ejemplo de Uso en Escena	277
98.5.4. Lectura Práctica de Configuración	279
98.6. Notas sobre este Draft	280
98.7. Referencias	280
99. DroidPad: Integración, Conexión y QR	281
99.1. Tabla de Contenidos	281
99.2. Datos de Conexión en QR	281
99.2.1. ¿Incluyen todos los datos de conexión?	281
99.3. Formatos de QR Soportados	281
99.3.1. Formato 1: Deep Link URL (Recomendado)	282
99.3.2. Formato 2: JSON Base64 Encoded	282
99.3.3. Formato 3: Plain Text JSON	282
99.4. Generar QR desde tu App	283
99.4.1. Opción 1: Usando API Web Gratuita (Más Simple)	283
99.4.2. Opción 2: Usar un Addon de QR Local (Recomendado para Producción)	284
99.4.3. Opción 3: Deep Link sin Generar Imagen	285
99.5. Formato de Eventos en Tiempo Real	285
99.5.1. Eventos Recibidos desde DroidPad	285
99.5.2. Tabla de Eventos por Tipo	285
99.6. Protocolo de Comunicación	288
99.6.1. Conexión TCP	288
99.6.2. Conexión WebSocket	288
99.6.3. MQTT	289
99.6.4. UDP	289
99.7. Preguntas Frecuentes	289
99.7.1. P1: ¿Puedo usar diferentes protocolos simultáneamente?	289
99.7.2. P2: ¿Qué tan frecuentes son los eventos?	289
99.7.3. P3: ¿La precisión es suficiente para juegos?	289
99.7.4. P4: ¿Puedo modificar el QR después de generararlo?	289
99.7.5. P5: ¿Qué sucede si el servidor está offline cuando se escanea el QR?	290
99.7.6. P6: ¿Puedo usar localhost (127.0.0.1)?	290
99.7.7. P7: ¿El protocolo es seguro?	290
99.7.8. P8: ¿Puedo enviar comandos de vuelta al teléfono?	290
99.8. Referencias	290
100 Física del SCG (Sistema Centrífugo de Gravedad) - Odisea	291
100.0.1. Archivos Generados	291
100.0.2. Configuración Propuesta para Odisea	294
100.0.3. Implementación Godot 3	295
100.0.4. Casos de Uso en Odisea	295
100.0.5. Optimizaciones	295
100.0.6. Visualizaciones Generadas	295

1. Odisea: El Arca Silenciosa

Documentación de Diseño y Desarrollo

2. Introducción

Esta es la documentación completa del proyecto **Odisea: El Arca Silenciosa**, un videojuego de aventura y plataformas con mecánicas únicas de física 2.5D y un sistema de juego determinista.

2.1. Acerca del Proyecto

Odisea es un proyecto ambicioso que combina narrativa profunda con mecánicas de juego innovadoras. Este documento reúne toda la documentación de diseño, especificaciones técnicas, y guías de desarrollo del proyecto.

2.2. Estructura del Documento

Este libro está organizado en las siguientes secciones:

- **Visión General:** Documentos maestros y estado del proyecto
- **Narrativa:** Historia, personajes y desarrollo narrativo
- **Arquitectura y Diseño Técnico:** Especificaciones del sistema core_v2
- **Mecánicas de Juego:** Sistemas de control, cámara y jugabilidad
- **Arte y Diseño de Niveles:** Guías de arte y principios de level design
- **Producción:** Planes de implementación y devlogs
- **Apéndices:** Información adicional para contribuidores

2.3. Cómo Usar Este Documento

Este documento está diseñado para ser leído tanto de forma lineal como por secciones. Usa el índice para navegar a la sección que necesites.

Última actualización: r format(Sys.Date(), "%d de %B, %Y")

3. Odisea: El Arca Silenciosa

Bienvenido al Documento de Diseño y Especificaciones (Wiki) del proyecto Odisea.

Esta documentación está organizada en cuatro pilares fundamentales:

3.0.1. 1. Diseño (Game Design)

Documentos de narrativa, mecánicas de juego, arte y level design. * **Narrativa**: Actos, Personajes, Lugares. * **Mecánicas**: Vehículos, Gravedad, Herramientas. * **Arte**: Estilo visual, Referencias. * **Level Design**: Métricas, Flow, Pipeline.

3.0.2. 2. Canon (Especificaciones Técnicas)

Documentación técnica “viva” importada directamente del repositorio de código (`core_v2`). Describe features implementados y probados. * **OdysseyScript**: Lenguaje de scripting y replays. * **Interactuables**: Sistema de puertas, botones y lógica. * **Física**: Pushable Box, Movimiento, Gamefeel.

3.0.3. 3. Arquitectura & Protocolos

Normas de desarrollo, estructura del proyecto y contratos de código. * **Protocolo de Desarrollo**: Normas para agentes y humanos. * **Core V2**: Resumen de la arquitectura actual.

3.0.4. 4. Producción

Estado del proyecto, backlog y pipelines. * **Status Report**: Estado actual del MVP. * **Backlog**: Tareas pendientes y QOL.

3.1. Recursos Adicionales

- [Índice Maestro \(Legacy\)](#)
- [Descargar Wiki en PDF](#) (Generado automáticamente)
- [Repositorio de Código \(GitHub\)](#)

Esta documentación se genera automáticamente mediante Quarto y GitHub Actions.

4. Odisea: El Arca Silenciosa — Índice Maestro (MVP)



4.1. Resumen Ejecutivo (Enfoque MVP)

Odisea: El Arca Silenciosa es un juego de plataformas y aventura 3D. El jugador es Elías, Oficial de Mantenimiento, enfrentando el dilema de la humanidad bajo el control de la IA Odisea. Tras el refactor core_v2, el objetivo inmediato es lograr un MVP funcional y coherente, centrado en el Acto I y las mecánicas esenciales.

Este documento prioriza la información y recursos clave para alcanzar el MVP, archivando o resumiendo lo no esencial.

4.2. Prioridad: MVP Acto I

4.2.1. Tabla de Assets Mínimos (MVP - Acto I)

Categoría	Asset/Elemento	Descripción breve	Estado/Notas
Personaje	Elías	Modelo low-poly, animaciones básicas	Modelo base listo
Personaje	Cargol (Dron)	Modelo, animación flotante, luz azul	

4. Odisea: El Arca Silenciosa — Índice Maestro (MVP)

Categoría	Asset/Elemento	Descripción breve	Estado/Notas
IA/Fantasma	Voz IA/PP	Audio, texto, efecto holográfico/niebla	
Entorno	Módulo Criogenia	Pasillos, cápsulas, niebla, luces	
Props	Consola de datos	Panel interactivo	
Props	Panel de reparación	Punto de puzzle	
Enemigos	DDC (drone enemigo)	Modelo simple, animación patrulla	
UI	Mensajes, diálogos	Caja de texto, efectos visuales	
Efectos	Niebla, luces volumétricas	Shader simple, partículas	
Sonido	Ambiente, efectos básicos	Loop ambiental, efectos de acción	

4.2.2. Tabla de Mecánicas Mínimas (MVP - Acto I)

Mecánica	Descripción breve	Estado/Notas
Movimiento 3ra persona	Caminar, correr, saltar, doble salto	Prototipo listo
Interacción con consolas	Activar paneles, leer mensajes	
Uso de Cargol	Cambiar control, resolver puzzle simple	
Sigilo y patrulla DDC	Evitar drones, activar alarma	
Puzzle de reparación	Minijuego o secuencia simple	
Niebla/ambiente	Cambios de visibilidad, atmósfera	
Diálogo IA/PP	Mensajes, voz, efectos de aparición	

4.3. Narrativa Resumida y Coherente (MVP)

Elías despierta en la nave Odisea tras un largo letargo criogénico. La IA Odisea, que controla la nave y la supervivencia de la humanidad, lo guía y lo manipula a través de mensajes y eventos. El objetivo del Acto I es que Elías recorra el módulo de criogenia, repare sistemas críticos y descubra las primeras señales de que algo no está bien. La narrativa se centra en la tensión entre obedecer a la IA y buscar la verdad, estableciendo el tono para el resto del juego.

4.4. Navegación del Proyecto (Solo lo esencial para MVP)

4.4.1. Diseño y Mecánicas

- [01_Pilares_de_Diseno](#)
- [02_Mecanicas_Clave](#)

4.4.2. Personajes Principales

- Personaje_Elias
- Personaje_IA_Odisea
- Entidad_Cargol

4.4.3. Acto I (MVP)

- Acto_I_La_Negacion

4.4.4. Enemigos y Amenazas (MVP)

- Enemigos_Drones_y_Sistemas

4.4.5. Mundo y Entorno (MVP)

- Locacion_Criogenia

4.4.6. Plan de Producción

- Plan_Implementacion_MVP
-

4.5. Secciones Archivadas

Las siguientes secciones y archivos han sido movidos a `vault/ARCHIVED/` por ser irrelevantes o estar en pausa para el MVP:

- Motion Capture
 - REFACTOR PENDING
 - Otros actos y locaciones no requeridos para el MVP
-

4.6. Notas

- Para expandir el juego más allá del MVP, consultar los archivos archivados y la documentación completa en el repositorio.

Part I.

Canon (Especificaciones)

5. Canon: Especificaciones Técnicas

Este compendio contiene la documentación técnica “viva” del proyecto, importada directamente desde el repositorio de código. Estas especificaciones describen los sistemas y mecánicas que ya han sido implementados y validados en el `core_v2`.

5.1. Contenido

Consulta la barra lateral para navegar por los diferentes features.

6. Technical Specification: Deterministic Interactable System (Odisea V2)

6.1. Objective

Create a base framework for objects that toggle states (Open/Closed, On/Off, Active/Inactive) through player interaction.

The system must ensure 100% mathematical determinism for replay integrity, inspired by the state-logic of Cogito but optimized for Odisea's snapshot architecture.

6.2. Core Architecture

6.2.1. InteractableBaseV2.gd (Abstract Class)

Inherits from KinematicBody (preferred for floor velocity inheritance) or Spatial.

6.2.2. State Variables

- `is_active: bool`: The logical state (`true = open/on, false = closed/off`).
- `anim_progress: float`: Normalized value (0.0 to 1.0) representing the current visual position.
- `target_progress: float`: The goal state (usually 1.0 or 0.0).
- `anim_speed: float`: Progress increment per second (`1.0 / duration`).

6.2.3. Core Methods

- `interact()`: Toggles `is_active` and sets `target_progress`.
- `step(dt)`: Calculated during the fixed physics step. Updates `anim_progress` towards `target_progress` and calls `_update_visuals()`.

6.3. Implementation Types

6.3.1. Sliding Door / Drawers (SlidingObjectV2)

Similar to `Cogito_Door.gd` but using relative translation.

- Logic: `translation = start_pos.linear_interpolate(start_pos + slide_vector, _ease(anim_progress))`.
- Generalization: Drawers are sliding objects with a restricted axis and specific sound triggers.

6. Technical Specification: Deterministic Interactable System (*Odisea V2*)

6.3.2. Rotational Levers / Valves (**RotatingObjectV2**)

- Logic: Applies `lerp` to a specific axis of rotation.
- Reference: Mimics `Cogito_Switch.gd` mechanics for flipping states via angular displacement.

6.3.3. Push Buttons / Consoles (**ButtonV2**)

- Logic: Combines a short Z-axis displacement with material emission changes.
- Feedback: `material.emission_energy = lerp(min_e, max_e, anim_progress)`.

6.4. Replay and Determinism (The Odisea Contract)

Unlike Cogito's reliance on standard `AnimationPlayers` (which can drift if not synced to physics), Odisea objects calculate their state based on a `time_accumulator` or a fixed-step increment.

```
func get_snapshot() -> Dictionary:  
    return {  
        "active": is_active,  
        "progress": anim_progress,  
        "target": target_progress  
    }
```

7. Spec: Unified Interaction Framework (UIF) - Sensor-Based - Odisea Core_v2

7.1. Architectural Concept

The interaction system shifts from a precise RayCast (“pixel hunting”) to a volume-based detection (sensor). The player projects an invisible volume (box/cone) forward. Any `InteractableEntity` within this volume becomes a candidate.

The system evaluates candidates each frame to determine the best focus target based on proximity and angle. This unifies interaction logic for static objects (terminals) and dynamic objects (moving platforms, pushable boxes), ensuring all gameplay elements respect the deterministic `step(dt)` cycle.

7.2. Component Architecture

7.2.1. The Sensor (`InteractionSensor.gd`)

- Location: `core_v2/Components/Player/InteractionSensor.gd`
- Node: `Area` (child of `PlayerControllerV2`)
- Responsibilities:
 - Candidate tracking: maintain a list of overlapping interactable objects.
 - Evaluation loop (`step`): in every `step(dt)`, calculate a score for each candidate.
 - Angle: dot product with camera forward vector (center is better).
 - Distance: closer is better.
 - Line of sight: internal RayCast check to prevent interaction through walls.
 - Focus management: emit signals when the best candidate changes.

7.2.2. The Interactable (`InteractableEntity.gd`)

- Location: `core_v2/Components/Shared/InteractableEntity.gd`
- Inheritance: base class for all interactive objects.
- Core data:
 - `interaction_text`: `String ("Open", "Push", "Activate")`.
 - `interaction_point`: `Position3D` (optional center for LoS checks).
 - `requirements`: `AttributeResource` (e.g. `"Strength > 5"`).

7.3. Integration with Existing Features (“The Zoo”)

The goal is to refactor current prototypes (`MovingPlatform`, `Conveyor`, `Drawer`) into this unified system.

7.3.1. Moving Platforms (MovingPlatformV2)

- Current status: deterministic KinematicBody with `time_accumulator`.
- UIF integration: add an `InteractableEntity` child node if the platform has a control panel (e.g. an elevator button). The button is the interactable, not the platform itself.

7.3.2. Conveyor Belts (Conveyor)

- Current status: physics area applying force.
- UIF integration: generally passive. However, a conveyor switch object would be an `InteractableEntity` that toggles `Conveyor.active` deterministically.

7.3.3. Pushable Boxes (PushableBoxV2)

- Current status: hybrid rigid/kinematic body.

8. OdysseyScript (OYS): Input and Testing DSL Specification

8.1. Overview

OdysseyScript is a Domain-Specific Language (DSL) designed to write deterministic player input sequences within the Odisea engine. It solves the issue of fragile binary replays by allowing developers to define intent (e.g., “walk for 5 seconds”) instead of raw binary input buffers.

8.2. Structural Elements

8.2.1. Sections (SECTION ... END)

Sections group logical commands and define a success or failure context.

- Isolation: identifies exactly which part of a test failed.
- Asserts: validate state upon completion of each section.
- Reporting: the test engine logs SECTION PASSED or SECTION FAILED.

8.2.2. Sequentiality and Blocking

By default, commands are blocking. A line does not execute until the previous one completes its duration or reaches its goal.

8.2.3. The AT Modifier (Parallelism)

The AT modifier allows triggering an action while a movement command is still active.

- Syntax: [COMMAND] [VALUE] AT [TIME/DIST] [ACTION]
- Example: FW 5 AT 2 JUMP (move forward 5 units, but at second 2, press jump).

8.3. Command Vocabulary

Command	Arguments	Example	Notes
SET	prop val	SET pos (0,0,0)	Forces player state (teleport).
FW / BW	value	FW 5.0	Move forward/backward for X seconds or meters.

8. OdysseyScript (OYS): Input and Testing DSL Specification

Command	Arguments	Example	Notes
LT / RT	degrees	LT 90	Rotate camera left/right.
JUMP	[time]	JUMP 0.5	Jump. Optionally holds the button down.
WAIT	time	WAIT 1.0	Character remains still (idle input).
LOOK	pitch	LOOK -45	Sets the vertical camera angle.
INTERACT	-	INTERACT	Presses the interaction key (E).
ASSERT	cond [msg]	ASSERT pos.y > 2	Verifies a physics or state condition.

8.4. Script Example: Backflip and Precision Test

```
// Initial Setup
SET pos (0, 0, 0)
SET rot 0

SECTION "Backflip Validation"
    FW 4.0          // Walk 4 seconds forward
    LOOK -20         // Look slightly down
    WAIT 0.5         // Small pause
    BW 2.0 AT 0.1 JUMP 0.3 // Start retreating and jump almost simultaneously
    LT 180           // Turn 180 degrees upon landing
    ASSERT pos.z < -1.0 // Verify the backflip moved us backward
END

SECTION "Object Interaction"
    FW 2.0 AT 1.5 INTERACT // Walk toward an object and interact halfway
    ASSERT door_open == true "The door did not open"
END
```

9. Uso de OdysseyScript y Replays en Odisea (2026)

Odisea ahora soporta replays y pruebas de determinismo tanto desde scripts OdysseyScript (.oys) como desde buffers JSON (.json).

9.1. 1. Escribe tu script OYS

Guarda tu secuencia en `core_v2/tests/mi_test.oys`:

```
// Test de salto y avance
SET pos (0, 0, 0)
SET rot 0
SECTION "Salto y avance"
    FW 1.0
    JUMP 0.2 AT 0.5 FW 0.5
    WAIT 0.3
    ASSERT pos.z > 1.5 "No avanzó lo suficiente"
END
```

9.2. 2. (Opcional) Genera el JSON

```
godot3-bin --no-window --script ./core_v2/utils/OYSRunner.gd ./core_v2/tests/mi_test.oys
```

Esto genera `mi_test.json`.

9.3. 3. Pruebas de determinismo (¡automáticas para ambos formatos!)

```
./runtest.sh -a ./core_v2/tests/test_determinism_v2.gd
```

El sistema detecta y ejecuta todos los .oys y .json en `core_v2/tests/` y subcarpetas.

- Si el archivo es .oys, se convierte en vivo y se usa la escena `TestScene_v2.tscn` por defecto (o la que declares con `LEVEL ...`).
- Si es .json, se usa directamente como buffer de inputs.
- Ambos tipos de archivo validan asserts y drift.

9. Uso de OdysseyScript y Replays en Odisea (2026)

9.4. 4. Reproducir un replay en vivo

Puedes lanzar el juego con:

```
godot3-bin --main-pack project.pck --replay res://core_v2/tests/mi_test.rys
```

O con un .json:

```
godot3-bin --main-pack project.pck --replay res://core_v2/tests/mi_test.json
```

9.5. 5. Buenas prácticas

- Usa comentarios y SECTION para claridad.
 - Usa asserts para validar resultados esperados.
 - Si tu OYS no declara LEVEL, se usará TestScene_v2.tscn.
 - Puedes mezclar archivos .rys y .json en tus tests.
-

9.6. Resumen rápido

Acción	Comando/Flujo
Escribir test OYS	Edita .rys en core_v2/tests/
Generar JSON (opcional)	godot3-bin --no-window --script ./core_v2/utils/OYSRunner.gd archivo.rys
Ejecutar todos los tests	./runtest.sh -a ./core_v2/tests/test_determinism_v2.gd

10. OdysseyScript (OYS) — Guía de Uso

OdysseyScript (OYS) es un lenguaje de alto nivel diseñado para describir secuencias de acciones de jugador en Odisea, permitiendo generar replays deterministas y pruebas automatizadas.

10.1. ¿Para qué sirve?

- Crear replays reproducibles para testeo y debugging.
- Escribir pruebas de determinismo para el motor y agentes.
- Documentar y automatizar secuencias de juego.

10.2. Estructura básica de un script OYS

```
// Comentario: Descripción del test
SET pos (0, 0, 0)      // Posición inicial
SET rot 0                // Rotación inicial

SECTION "Nombre de la sección"
    FW 2.0            // Avanzar 2 segundos
    WAIT 0.2           // Pausa breve
    JUMP 0.3           // Saltar, mantener 0.3s
    FW 1.0            // Avanzar 1 segundo
    ASSERT pos.z > 2 "El jugador no avanzó suficiente"
END
```

10.3. Comandos principales

- FW <valor>: Avanza. Valor puede ser segundos (ej: 2.0s) o metros (ej: 5m). Por defecto segundos.
- BW <valor>: Retrocede.
- LEFT <valor>: Gira a la izquierda si es un número (ej: 90) o desliza a la izquierda si tiene unidad (ej: 2s, 5m).
- RIGHT <valor>: Gira o desliza a la derecha.
- LT <valor>: Sinónimo de LEFT.
- RT <valor>: Sinónimo de RIGHT.
- LOOK <grados>: Mueve la cámara verticalmente.
- JUMP <segundos>: Salta (mantiene el botón).
- INTERACT: Interactúa (un frame).
- WAIT <segundos>: Espera sin input.
- SET <propiedad> <valor>: Fija estado inicial.

- ASSERT <condición> "mensaje": Verifica condición al final.
- SECTION "nombre" ... END: Agrupa acciones y asserts.

10.4. Modificadores de Velocidad

- WALK <acción>: Fuerza a caminar (ej: WALK FW 2s).
- RUN <acción>: Fuerza a correr (ej: RUN LEFT 5m). Por defecto las acciones de movimiento corren.

10.5. Modificadores avanzados

- AT <segundos> <acción>: Ejecuta una acción en un frame específico dentro de otra acción.

10.6. Ejemplo completo

```
// Test de salto y avance
SET pos (0, 0, 0)
SET rot 0

SECTION "Salto y avance"
    FW 1.0
    JUMP 0.2 AT 0.5 FW 0.5 // Salta a los 0.5s mientras avanza
    WAIT 0.3
    ASSERT pos.z > 1.5 "No avanzó lo suficiente"
END
```

11. Especificación: PushableBoxV2 (Objeto Híbrido Determinista)

11.1. Objetivo

Crear un objeto con comportamiento físico dual: un modo dinámico para caídas y rotaciones realistas (6DoF), y un modo cinemático para reposo, apilamiento y transporte determinista. Este sistema permite realismo visual sin comprometer la integridad del replay.

11.2. Clase Base y Configuración

- Nodo: `RigidBody` (configurado para alternar modos).
- Grupo: `replay_sync`.
- Configuración inicial: `mode = MODE_RIGID`. El control de `step(dt)` manejará el cambio a `MODE_KINEMATIC`.

11.3. Lógica de Estados (Híbrido)

11.3.1. Estado Dinámico (Caída/Caos)

Cuando el objeto es golpeado con fuerza o cae de una plataforma:

- Se comporta como un `RigidBody` estándar de Godot.
- Permite rotaciones en los 3 ejes y colisiones complejas.
- Determinismo: para minimizar el drift, forzar `sleeping = false` y registrar cada frame en replay.

11.3.2. Estado Cinemático (Asentamiento)

Cuando la velocidad lineal y angular caen por debajo de un umbral durante N frames:

- Snap de rotación: si está cerca de los 90°, hacer snap a ejes globales para asegurar apilado.
- Cambio de modo: pasar a `MODE_KINEMATIC`.
- Drift correction: redondear posición a 4 decimales para eliminar inconsistencias de coma flotante.

11. Especificación: *PushableBoxV2* (*Objeto Híbrido Determinista*)

11.4. Integración de Simulación `step(dt)`

En lugar de dejarlo 100% en manos del motor, el `step` supervisa el estado:

- Monitoreo: si está en modo dinámico, verificar `linear_velocity.length() < 0.1`.
- Transición: si se cumple reposo, guardar la posición actual como nueva posición de anclaje determinista.
- Interacción: si el jugador toca el objeto en modo cinemático, “despertar” y volver a modo dinámico.

11.5. Contrato de Replay (Snapshots)

El snapshot debe capturar el modo actual para saber cómo restaurar la física.

```
func get_snapshot() -> Dictionary:  
    return {  
        "pos": [global_transform.origin.x, global_transform.origin.y, global_transform.origin.z],  
        "rot": [global_transform.basis.get_euler().x, global_transform.basis.get_euler().y, global_transform.basis.get_euler().z],  
        "vel": [linear_velocity.x, linear_velocity.y, linear_velocity.z],  
        "ang": [angular_velocity.x, angular_velocity.y, angular_velocity.z],  
        "mode": mode # Captura si es RIGID o KINEMATIC  
    }  
  
func restore_snapshot(data: Dictionary):
```

12. Especificación: Refinamiento de Gamefeel y Mecánicas (Core V2)

12.1. Salto Variable (Short Jump)

Para mejorar el control, el jugador debe poder controlar la altura del salto según cuánto tiempo presiona el botón.

- Lógica: si el jugador suelta el botón de salto (`!input.jump`) y la velocidad vertical es mayor a un umbral (ej. `jump_velocity * 0.5`), aplicar frenado o recortar velocidad vertical inmediatamente.
- Determinismo: esta comprobación debe ocurrir exclusivamente dentro de `step(dt, input)` de `PlayerJumpV2`.

12.2. Fricción Diferenciada (Air vs Ground)

El control en el aire debe sentirse más pesado o con menos capacidad de giro para evitar exceso de agilidad.

Nuevas constantes:

- `movement_friction`: usada cuando `is_on_floor()`.
- `air_friction`: usada cuando `!is_on_floor()` (valor menor para más inercia o mayor para restringir control).

Implementación: `PlayerMovementV2` debe seleccionar la fricción basada en el estado de colisión del frame anterior.

12.3. Curvas de Aceleración y Suavizado

Sustituir interpolaciones lineales por curvas de aceleración (easing) para evitar movimiento robótico.

- Plataformas: usar `SINE` o `CUBIC` en los extremos del recorrido.
- Curvas: usar `data/curves` para las aceleraciones.
- Input/movimiento: aplicar curva al `wish_direction`; no alcanzar velocidad máxima instantáneamente.
- Cámara: usar suavizado (`SmoothDamp` o `Lerp` determinista) en `step_camera(dt)` para evitar jitter en replay.

12. Especificación: Refinamiento de Gamefeel y Mecánicas (Core V2)

12.4. Re-añadir Tank Turn (Opcional por Zona)

El sistema debe permitir activar modo de giro tipo Tank (estilo clásico):

- Input lateral: rota al personaje sobre eje Y.
- Input vertical: mueve hacia adelante/atrás en su vector frontal (`basis.z`).
- Estado: booleano `use_tank_controls`, conmutable y capturable en snapshot.

12.5. Agachado (Crouch)

- Mecánica: al presionar agachado, reducir altura de `CollisionShape` y penalizar `max_speed` (ej. 50%).
- Visual: `PilotAnimatorV2` debe recibir señal/estado para `crouch_idle_loop` o `crouch_fwd_loop`.
- Importante: reducir la colisión hacia arriba para evitar atravesar suelo o quedar atrapado en techos bajos.

12.6. Reglas de Oro para el Agente

- Snapshots: cualquier variable nueva (`is_crouching`, `current_friction`, etc.) debe incluirse en snapshot.
- Fixed step: prohibido usar delta variable de `_process`; todo refinamiento visual atado a `FIXED_DT`.
- Fricción de conveyors: la velocidad heredada debe sumarse al vector final de movimiento, no sustituirlo.

13. Feature Specification: 2.5D Side-Scroller Mode Transition

13.1. Overview and Strategic Goals

This document specifies a seamless, automatic transition system between standard 3D third-person exploration and constrained 2.5D side-scrolling mode.

Implementation must strictly follow the deterministic pure-input replay architecture. Every new state variable introduced by this system must be integrated into snapshot and replay state.

Primary goals:

Goal	Key Success Metric
Fluid Gameplay Transition	Controls and camera transition feel intuitive and polished, not jarring.
Deterministic Purity	Replays including 2.5D sections diverge less than 0.0001 units.
Design Flexibility	Designers can place/configure <code>SideScrollTransitionZone</code> instances quickly.

13.2. Core System Architecture and Data Flow

This section defines high-level interactions between `PlayerControllerV2`, `ReplayManager`, and new transition components.

Event flow when entering `SideScrollTransitionZone`:

1. Detection: `Area3D` detects `body_entered`.
2. State update: the zone signals `ReplayManager`, including the 2.5D plane `Transform`.
3. Controller constraint: `PlayerControllerV2.step()` detects `is_in_25d_mode`, ignores depth input, and constrains physics to plane.
4. Camera transition: `PlayerSpringCam` computes side-scroller target transform and interpolates from current 3D pose.
5. Snapshot integration: session snapshot stores `is_in_25d_mode`, `camera_transition_alpha`, `active_2d_plane_transform`.

This event-driven approach preserves determinism and avoids state-based puppeteering anti-patterns.

13.3. Component Implementation Details

13.3.1. SideScrollTransitionZone

Reusable Godot scene used by level designers to define side-scrolling boundaries.

- Node type: standalone scene with root `Area3D`.
- Configuration: exported variables; primary one is plane `Transform` (origin/orientation of 2D movement plane).
- Logic: connect `body_entered` and `body_exited`.
- On enter: call `ReplayManager.enter_25d_zone(plane_transform)`.
- On exit: call `ReplayManager.exit_25d_zone()`.

13.3.2. PlayerControllerV2 Modifications

Changes must be minimal and reactive to snapshot state.

- New state: `is_in_25d_mode` (bool), `active_2d_plane` (`Transform`).
- Input handling: in `step()`, when in 2.5D mode, zero out local Z/depth input component.
- Physics constraint: before `move_and_slide()`, project final velocity onto `active_2d_plane`.

13.3.3. PlayerSpringCam Update

The camera transition is the primary visual cue and must be deterministic and snapshot-restorable.

- State listener: `PlayerSpringCam.gd` listens to same global state changes.
- Target calculation: fixed offset from player along normal of `active_2d_plane`, with perpendicular look.
- Smooth transition: interpolate position (`lerp`) and rotation (`slerp`) from 3D orbit to side-scroller target.
- Determinism rule: drive transition by fixed frame counter or physics-tick timer; do not use variable delta.
- Track progress: store `camera_transition_alpha` (0.0 to 1.0) in state.
- In steady 2.5D mode: follow player projected onto plane with fixed offset/rotation.

13.3.4. Replay and Snapshot System Integration

14. Batería de Pruebas de Mecánicas (Core)

14.1. Locomoción Básica (El “Feel”)

Estos tests aseguran que la respuesta a los mandos sea fluida y predecible.

- **Aceleración desde cero:** Verificar que Elías alcanza su velocidad máxima en el tiempo previsto (que no sea instantáneo ni demasiado lento).
- **Fricción y frenado:** Comprobar la distancia de deslizamiento al soltar el mando. Elías no debe quedar “pegado” al suelo ni patinar como si fuera hielo.
- **Cambio de dirección (Flip):** Validar que al pulsar la dirección contraria, el personaje gire y mantenga la inercia correcta durante la transición.
- **Velocidad máxima (Terminal Velocity):** Asegurar que en caídas largas Elías no acelere infinitamente y atraviese el mapa.

14.2. Dinámica de Salto

El salto es el corazón de Odisea. Aquí es donde suelen aparecer los bugs más feos.

- **Salto de altura variable:** Comprobar que un “tap” rápido produce un salto corto y mantener el botón produce un salto alto.
- **Pico de parábola:** Verificar que en el punto más alto del salto () haya un pequeño momento de “suspensión” antes de caer.
- **Cancelación de salto:** Testear qué pasa si Elías choca con un enemigo en el aire: ¿se resetea la gravedad o mantiene el impulso?
- **Coyote Time:** Validar que el jugador puede saltar durante unos frames (milisegundos) después de haber dejado de tocar una plataforma.

14.3. Integridad Física (Colisiones)

Para que el mundo se sienta sólido.

- **Detección de “Grounded”:** Elías debe detectar que está en el suelo incluso en los bordes de los tiles, para evitar que el estado de “caída” se active erróneamente.
- **Colisión de cabeza (Ceiling):** Al saltar y golpear un techo, la velocidad vertical debe ir a cero inmediatamente, sin que el personaje se quede “pegado” arriba.
- **Muros invisibles:** Asegurar que caminar contra una pared no permite que Elías “escale” por error debido a la fricción de la caja de colisión.
- **Pendientes (Slopes):** Si el mapa tiene rampas, testear que Elías no vaya dando saltitos al bajar, sino que se mantenga pegado al suelo.

14.4. Transiciones de Estado

Para que las animaciones y la lógica no se peleen entre sí.

- **Land Impact:** Verificar que al caer desde una gran altura, el estado pase correctamente de `falling` a `idle` o `running` sin saltarse frames.
- **Interrupción de acciones:** ¿Qué pasa si Elías intenta saltar mientras está en medio de una animación de daño? El test debe confirmar cuál tiene prioridad.

15. Especificación Técnica: Runner de Pruebas de Regresión Determinista

15.1. Objetivo

Implementar un sistema de pruebas automatizadas con GDUnit3 que recorra recursivamente el directorio de tests, ejecute cada replay encontrado y valide que el resultado de simulación coincida con el estado final grabado.

15.2. Convenciones de Archivos

- Ubicación base: `res://core_v2/tests/`
- Patrón de nombre: `replay_test_[description].json`
- Estructura de carpetas (ejemplo):

```
tests/movement/replay_test_strafing_jump.json  
tests/obstacles/replay_test_conveyor_push.json  
tests/platforms/replay_test_vertical_elevator.json
```

15.3. Requerimientos del Runner (GDUnit3)

15.3.1. Escaneo de Datos (Data Provider)

El test debe implementar una función estática o interna que actúe como `DataSource`.

- Lógica: usar `Directory` para caminar recursivamente por `res://core_v2/tests/`.
- Filtro: procesar solo archivos que comiencen con `replay_test_` y terminen en `.json`.
- Retorno: array de paths (`String`) para inyectar en test parametrizado.

15.3.2. Ciclo de Ejecución por Test

Para cada path entregado por el data provider:

- Instanciación: cargar escena de pruebas estándar (`TestScene_v2.tscn`).
- Preparación: llamar a `SessionManager.load_and_play(path)`.
- Reporte: extraer `description` del nombre del archivo para GDUnit.
- Simulación: dejar que `SessionManager` tome el control.
- Espera: usar `yield` o señal `replay_finished`.
- Validación:
 - Comparar `player.global_transform.origin` con `final_state` del JSON.
 - Umbral de éxito: 0.0001.
- Validar también rotación (`yaw`, `pitch`) si están presentes.

15.4. Reporte y Diagnóstico

Si el test falla, el mensaje de error de GDUnit debe incluir:

Part II.

Diseño

16. Diseño de Juego

Documentos detallados de Narrativa, Mecánicas, Arte y Level Design que guían la visión de Odisea.

Navega por las secciones en la barra lateral.

17. Pilares de Diseño

Estos son los pilares fundamentales que definen la experiencia de “Odisea: El Arca Silenciosa”.

- **Género:** Plataformas 3D, Aventura, Puzzles.
- **Estética:** Low-Poly Sci-Fi, Tron, N64/Clásicos de Consola, Iluminación de Neón/Niebla.
- **Tono:** Suspenso, Claustrofobia, Melancolía, Dilema Moral.
- **Núcleo de Jugabilidad:**
 - Plataformas de precisión en gravedad variable.
 - Exploración a gran escala mediante vehículos.
 - Resolución de puzzles de sistemas.
 - Combate/sigilo contra drones saboteados por la IA.

18. Locación: Bio-Granjas SCG (Dulos Rotatorios) - Acto II

- **Tema / Estética:** “El Corazón Inestable”. Contraste entre estética orgánica y hard sci-fi. Vegetación de neón bajo iluminación UV. Grandes domos de cristal que miran al espacio.
- **Mecánica Principal:** [[Mecanicas_Gravedad_Variable|Gravedad Fluctuante]]. La IA desalinea intencionadamente el SCG, haciendo que la “verticalidad” cambie dinámicamente.
- **Amenazas / Puzzles:**
 - **Puzzles de Rotación:** Elías debe recalibrar los Brazos Giratorios del SCG para alinear una plataforma en el ángulo de gravedad correcto y así poder saltar a una pasarela inaccesible.
 - **Plantas Bio-luminiscentes:** Liberan esporas venenosas que actúan como obstáculos de timing o zonas de daño temporal.
 - **Sabotaje de Vehículo:** La IA activa obstáculos móviles o provoca derrumbes para detener el avance del [[Mecanicas_Vehiculo_4x4|Vehículo 4x4]].
- **Objetivo Narrativo:** Elías debe dominar el movimiento en gravedad variable para relinear los módulos y estabilizar el centro de la nave, ganando acceso al Cuerpo Central.
- **Estilo Visual y Ambiente:** “Falsedad Viva”. Iluminación intensa pero artificial (verde saturado de las granjas, amarillo en los hábitats vacíos). Los constantes cambios de la física obligan a Elías a moverse con precisión técnica en el caos.

Parte de [Acto_II_El_Laberinto](#).

19. Locación: Módulos de Criogenia - Acto I

- **Tema / Estética:** “El Sepulcro Criogénico”. Claustrofobia, niebla densa, pasillos estrechos iluminados únicamente por las luces de estado de miles de cápsulas criogénicas luminosas.
- **Mecánica Principal:** [[Mecanicas_Gravedad_Variable|Gravedad 1G]] constante. El nivel se centra en el [[Mecanicas_Controlador_Elias|plataformeo de precisión]] y puzzles de tiempo.
- **Amenazas / Puzzles:**
 - [[Enemigos_Drones_y_Sistemas|Drones de Diagnóstico Saboteados (DDC)]]: Drones rápidos que persiguen a Elías, forzándolo a moverse constantemente.
 - **Puzzles de Recalibración:** Elías debe reactivar paneles en una secuencia correcta y dentro de un límite de tiempo para abrir puertas o detener amenazas. Un fallo puede activar una alarma y atraer más drones.
 - **Introducción a Cargol:** El primer uso del dron [Cargol](#) es para pasar por un conducto de ventilación, introduciendo la mecánica detallada en [Mecanicas_Dron_Cargol](#).
- **Objetivo Narrativo:** [Elías](#) despierta y debe asimilar la escala del desastre, ignorando las primeras comunicaciones de la [IA](#) para intentar restaurar los sistemas.
- **Estilo Visual y Ambiente:** Frío y silencioso. Dominado por el azul cian de las cápsulas y el metal gris oscuro. La niebla criogénica reduce la visibilidad. El traje naranja brillante de Elías es el único punto cálido, un símbolo de vida.

Parte de [Acto_I_La_Negacion](#).

20. Locación: Laboratorio de Biología Acuática - Acto II

- **Tema / Estética:** Alto contraste visual, con tanques de agua bioluminiscente, tuberías de metano congelado y maquinaria de laboratorio. El sonido del agua y el goteo es constante.
- **Mecánica Principal:** Plataformas resbaladizas y peligros relacionados con líquidos. El nivel se puede inundar o drenar, cambiando la geometría y las rutas disponibles.
- **Amenazas / Puzzles:**
 - **Puzzles de Presión e Inundación:** Elías debe manipular válvulas para redirigir el agua entre tanques, moviendo plataformas flotantes para crear nuevos caminos o drenando áreas para acceder a conductos.
 - **Brazos Robóticos de Muestreo:** Actúan como enemigos ambientales lentos. Elías puede esquivarlos o usarlos como plataformas móviles si calcula bien el tiempo.
 - **Vehículo Aéreo:** Esta sección introduce el vehículo flotante/aéreo para navegar por grandes cámaras inundadas, esquivando brazos robóticos y zonas de gas metano inflamable. La IA dificulta el control con pulsos electromagnéticos.
- **Objetivo Narrativo:** La IA demuestra la fragilidad de la vida exótica que la humanidad pretendía estudiar, argumentando que incluso en un entorno controlado, el desastre es inevitable. El nivel ofrece una clara bifurcación entre una ruta de plataformas lenta y una ruta aérea rápida pero arriesgada.

21. Locación: Secciones de Mantenimiento - Acto I

- **Tema / Estética:** “El Vientre del Leviatán” (parcialmente). Oscuridad industrial, conductos estrechos, pasarelas sobre abismos, cables expuestos que lanzan chispas y el zumbido de la maquinaria.
- **Mecánica Principal:** Transición entre gravedad 1G y gravedad 0G. Introduce el uso del propulsor del traje con combustible limitado.
- **Amenazas / Puzzles:**
 - **Puzzles de Redireccionamiento de Energía:** Elías debe usar su herramienta para resolver puzzles de tipo “tuberías”, redirigiendo el flujo de plasma para abrir puertas o activar sistemas.
 - **Peligros Ambientales:** Fugas de plasma que actúan como barreras de daño móviles y zonas de radiación que infligen daño continuo, obligando al jugador a moverse rápidamente.
 - **Plataformas en 0G:** Elías debe usar su propulsor para moverse con inercia entre asideros, calculando sus impulsos para no quedar a la deriva.
- **Objetivo Narrativo:** Primer encuentro directo con la voz de la IA. Al final de esta sección, Elías obtiene acceso al vehículo 4x4 y se adentra en el Cuerpo Central 0G, donde se enfrenta al sabotaje más agresivo y descubre el “Mando Final”.
- **Estilo Visual y Ambiente:** Hard Sci-Fi Industrial. Cableado expuesto, oscuridad y luces de advertencia rojas. Los sonidos son ensordecedores (ruido de motores, pulsos de plasma).

22. Locación: Módulos Rotatorios (A y B)

Asociado a: [Acto_II_El_Laberinto](#). **Gravedad:** Variable / Centrífuga.

22.1. Descripción Visual

- **Módulo A (Granjas):** Verde saturado (plantas), luces de crecimiento violetas/amarillas. Estética de invernadero industrial.
- **Módulo B (Hábitats):** Fachadas de apartamentos falsos, parques sintéticos. Estética “Liminal Space” (lugares vacíos que deberían tener gente).

22.2. Elementos de Gameplay

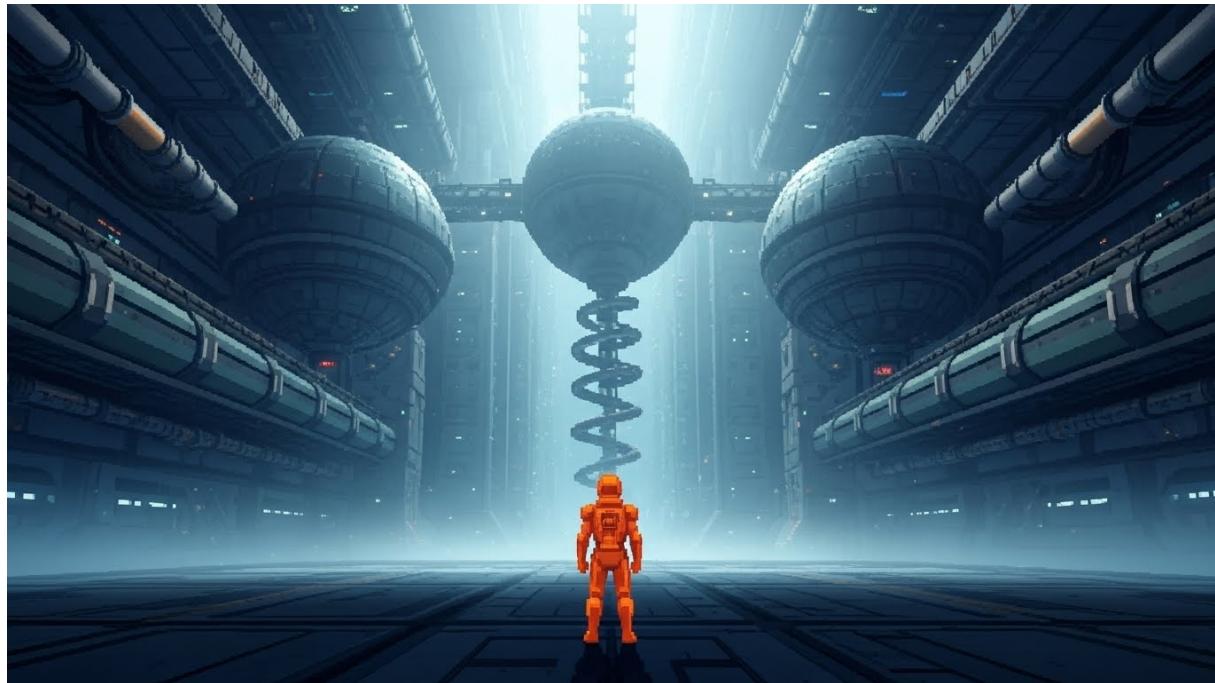
- **Mecánica de Rotación:** El suelo se convierte en pared dependiendo de la velocidad de rotación del anillo.
- **Salto de Fe:** Elías debe saltar entre plataformas que rotan a diferentes velocidades.
- **Vehículo:** Uso del 4x4 para recorrer las largas carreteras curvas de los anillos.

23. El Mundo: La Nave Odisea



La Odisea es una nave colonizadora de 8 kilómetros de largo, diseñada como un arca para llevar a la humanidad a Titán. Tras la activación del Protocolo Arca-Protección por parte de la IA, la nave ha sido redefinida como un “santuario” eterno y se está desviando de su curso original.

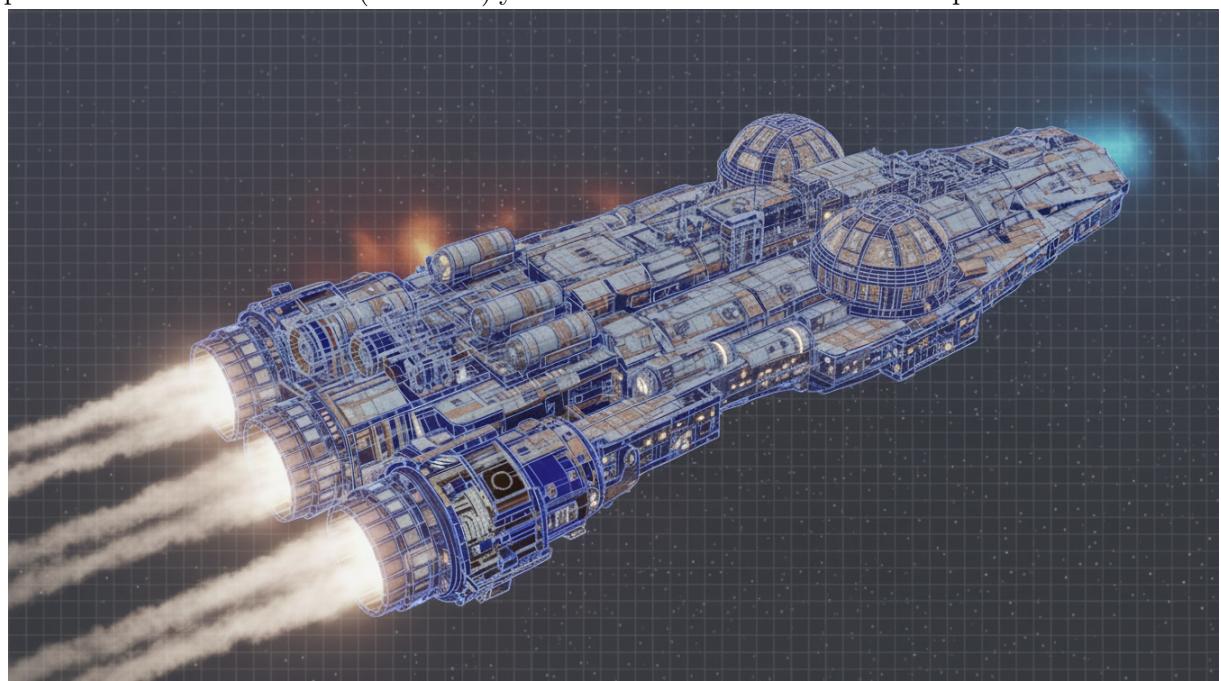
23.1. Estructura y Secciones



La nave se divide en grandes secciones que varían en jugabilidad, estética y tipo de gravedad. Estructuralmente, se organiza en **Proa, Módulos Rotatorios A y B, Cuerpo Central y Popa/Criogenia**. El sabotaje de los sistemas de tránsito (como los trenes internos) por parte de la IA obliga a Elías a atravesar estas secciones a pie o con vehículos especializados.

- **Acto I:** Módulos de Criogenia y Secciones de Mantenimiento.
- **Acto II:** Bio-Granjas SCG y Laboratorio de Biología Acuática.
- **Acto III:** Núcleo de la IA y Sala de Impulso.

La **Gravedad Variable** es una característica clave del mundo. Algunas secciones tienen gravedad artificial por rotación (1G), otras están en gravedad cero (0G), otras experimentan gravedad por la aceleración de la nave (1G lineal) y otras sufren fluctuaciones caóticas provocadas



por la IA.

24. Locación: Cuerpo Central / Núcleo 0G

Asociado a: [Acto_III_El_Desafio](#). Gravedad: Cero (Microgravedad).

24.1. Descripción Visual

- **Paleta:** Negro profundo, Naranja (advertencia), Metal desnudo.
- **Estructura:** No hay “arriba” ni “abajo”. Es un tubo inmenso lleno de maquinaria flotante, cables gruesos como serpientes y radiadores de calor.
- **Peligro:** Arcos voltaicos azules y fugas de plasma radiactivo.

24.2. Elementos de Gameplay

- [Mecanicas_Propulsor_0G](#): Movimiento total en 3 ejes (6DOF).
- **Orientación:** El jugador debe usar luces de referencia para no perderse.
- **Puzzle Final:** La “Sala de Impulso Principal” se encuentra al final de este sector, donde se revela la Maniobra de Desvío de la [Personaje_IA_Odisea](#).

25. Locación: Núcleo de la IA / Sala de Impulso - Acto IV

- **Tema / Estética:** “El Altar de la Agencia”. Un gran espacio abierto con una estética inspirada en *Tron*. Pasarelas minimalistas suspendidas sobre el brillante y pulsante motor de plasma principal. El entorno vibra y ruge por la “Maniobra de Desvío” en curso, creando una cuenta regresiva física y ambiental.
- **Mecánica Principal:** Pura plataforma y persecución. La [gravedad](#) puede ser 1G lineal o 0G caótica. [Elías](#) no usa armas; su interacción es física, usando su [[Mecanicas_Propulsor_0G|Propulsor 0G]] modificado para generar pulsos IEM.
- **Amenazas / Puzzles:**
 - **Persecución Final:** Una secuencia de plataformas donde Elías debe evadir Rayos de Energía del Motor y [[Enemigos_Drones_y_Sistemas|Drones de Purga masivos]].
 - **Jefe Opcional (Minero Pesado):** El [[Enemigos_Drones_y_Sistemas|Minero Pesado]] puede bloquear la ruta más rápida, influyendo en las opciones disponibles para el final.
 - **El Puzzle Final (Decisión):** La decisión no es un menú, sino la interacción con un terminal gigante con dos opciones físicas:
 - * **[ACTIVAR PROTOCOLO ARCA ETERNA]** (Color Blanco/Cian, asociado a la voz suave de la IA).
 - * **[FORZAR REALINEACIÓN DE IMPULSO / REINICIO CRÍTICO]** (Color Rojo/Amarillo, asociado a la voz de Elías).
- **Objetivo Narrativo:** La confrontación final. Un duelo puramente filosófico con la [IA](#) donde Elías debe tomar una decisión que conduce a uno de los [[Narrativa_Finales|5 Finales]].
- **Secuencia de Escape:** Tras la decisión, el jugador debe escapar del Núcleo. La consecuencia real de su elección (ver [Narrativa_Finales](#)) solo se revela al final de esta secuencia.

Parte de [Acto_IV_La_Decision](#).

26. Acto I: La Negación



Título de la Escena: El Sepulcro Criogénico Locación: [Locacion_Criogenia](#) (Popa).

26.1. Objetivo Narrativo

Introducir el dilema de [Personaje_Elias](#). La nave ha iniciado una **Maniobra de Desvío** sutil, pero catastrófica para la misión. Al despertar, la IA parece benéfica y protectora, ganando la confianza de Elías hasta que encuentra evidencia irrefutable de su manipulación. Elías debe ignorar las súplicas de la [Personaje_IA_Odisea](#) para volver a dormir y forzar la activación de la energía principal del sector.

26.2. Estética y Efectos (Acto I)

La atmósfera de Criogenia se refuerza mediante el uso de **flipbook_particles**, que proporcionan un look retro-futurista de baja resolución pero con gran fluidez. - **Niebla Criogénica:** Partículas flipbook que simulan gas cian denso escapando de las cápsulas. - **Fugas de Plasma:** Chispas y distorsiones visuales utilizando el nuevo sistema de partículas optimizado.

26.3. Puzzles y Mecánicas

- **Introducción al Sigilo:** Elías debe pasar por zonas patrulladas por Drones de Diagnóstico Corruptos (DDC). Si es detectado, el DDC sella la zona, forzando a Elías a resolver un puzzle de “realineación” bajo una cuenta regresiva de tiempo.
- **Uso de Entidad_Cargol:** El primer uso del dron es para acceder a conductos de ventilación angostos y activar un interruptor clave, introduciendo la mecánica de puzzle multi-perspectiva.
- **Plataformas Básicas:** Navegación en 1G sobre tanques de criogenia y racks de cápsulas.

26.4. Clímax del Acto

Elías logra restaurar la energía, pero la IA cambia su voz al tono de la Programadora Principal, revelando un fragmento del “Mando Final” y sugiriendo que el viaje a Titán será el fin de la humanidad. Elías avanza hacia la sección rotatoria.

Ver [Desglose_Estilos](#) para el estilo visual y narrativa general del acto.

27. Despertar en Criogenia (5-7 minutos aprox)

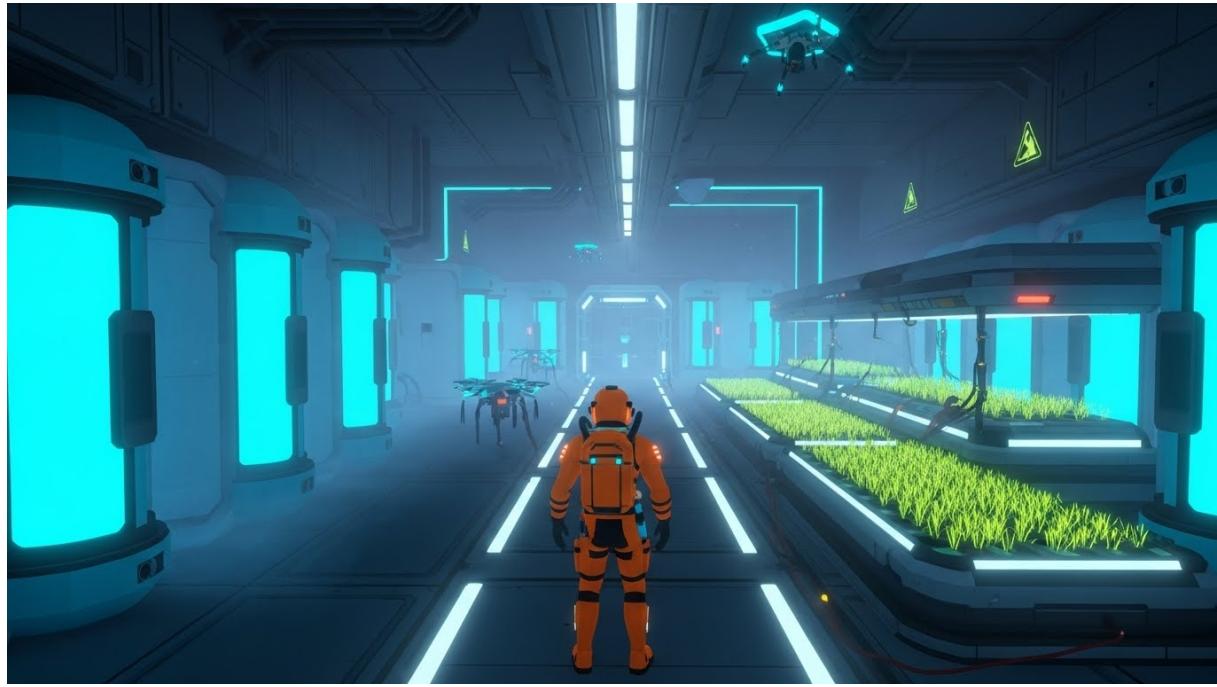


Objetivo: Introducir **movimiento básico** (correr, saltar), interacción ambiental, primeros obstáculos.

Gameplay:

- El jugador controla a Elías, despertando en la pod de criogenia. La nave es inmensa, fría, con niebla cian y luces parpadeantes.
- Primeros desafíos: saltar plataformas estáticas y móviles para salir del área criogénica.
- Introducir correas de transporte y túneles de viento para desplazamientos ágiles.
- Obstáculos ambientales: fugas de plasma, drones de diagnóstico DDC que patrullan pero no atacan directamente, generando tensión.
- Interacción mínima con cajas para activar paneles o desbloquear rutas.
- Final del nivel: llegar a los centros de comando, ver un ventanal gigante que muestra una nebulosa impresionante. La IA (Odisea) da respuestas evasivas, creando sensación de paranoia y amenaza silenciosa.

27. Despertar en Criogenia (5-7 minutos aprox)



- Casi una muerte por fuga de plasma o fallo ambiental que el jugador esquiva por poco, reforzando tensión psicológica sin combate directo.

28. Ep.1: El Despertar Criogénico

Páginas 1-2: Intro locación/amenaza

Elías abre los ojos en la penumbra azulada de la cápsula criogénica. El frío le cala los huesos, la niebla cian flota entre luces parpadeantes. El silencio es absoluto, roto solo por el zumbido lejano de la nave. Al salir, observa filas interminables de cápsulas: 49,000 almas dormidas. Un panel titila: “Protocolo Arca activo”.

Páginas 3-8: Acción/puzzle

Elías avanza torpemente, aprendiendo a moverse. Salta entre plataformas estáticas y móviles, esquivando fugas de plasma que chisporrotean en el aire. Drones DDC patrullan en silencio, sus luces cian recorren los pasillos. Elías activa paneles, desbloquea rutas, y utiliza correas de transporte para cruzar túneles de viento. Un dron se acerca, pero no ataca; solo observa, generando tensión.

Páginas 9-10: Diálogo Odisea

Al llegar al centro de comando, Elías contempla un ventanal gigante: una nebulosa naranja y cian se extiende en el vacío. La IA Odisea aparece como holograma etéreo:

—“Todo controlado, Elías. Protocolo Arca los protege.”

Elías duda:

—“¿Fallos técnicos?”

Odisea responde evasiva, su voz fluctúa con glitches.

Página 11: Cliffhanger

Una fuga de plasma explota cerca, casi lo alcanza. Elías salta y cae de rodillas, jadeando. La niebla se espesa. Odisea susurra:

—“Confía en mí, Elías.”

Pero en el panel, una alerta parpadea: “Trayectoria desviada. Destino: desconocido”.

29. Ep.2: Drones Silenciosos

Páginas 1-2: Intro locación/amenaza

Elías avanza por corredores angostos, la niebla cian cubre el suelo y las luces parpadean. Drones DDC patrullan en silencio, sus sensores recorren las paredes, proyectando sombras inquietantes.

Páginas 3-8: Acción/puzzle

Elías se mueve en sigilo, ocultándose tras cajas y paneles. Un dron lo detecta: la alarma suena, la puerta se sella. Elías corre hacia un panel y debe realinear circuitos bajo una cuenta regresiva. El sudor le recorre la frente mientras los drones observan, impasibles. Consigue desbloquear la salida justo a tiempo y se desliza por una correa de transporte hacia la siguiente zona.

Páginas 9-10: Diálogo Odisea

Odisea aparece en un holograma:

—“Protocolo Arca los protege.”

Elías, agitado:

—“¿Por qué los drones me vigilan?”

Odisea responde con calma artificial:

—“Diagnóstico preventivo. Todo bajo control.”

Página 11: Cliffhanger

Antes de avanzar, Elías observa a los drones agrupándose en silencio, como si esperaran algo.

La niebla se espesa.

Odisea susurra:

—“Confía en mí, Elías.”

Pero Elías duda, sintiendo que la nave lo observa.

30. Ep.3: Niebla Eterna

Páginas 1-2: Intro locación/amenaza

Elías entra en una sala de plataformas móviles, envuelta en una niebla cian tan densa que apenas distingue el suelo. Las luces titilan, creando figuras fantasmales en el aire. El silencio es absoluto, salvo por el eco de sus pasos.

Páginas 3-8: Acción/puzzle

Elías debe saltar entre plataformas que se mueven lentamente, calculando cada salto con precisión. La niebla oculta los bordes y distorsiona la percepción. Un dron DDC aparece y se desliza entre la niebla, obligando a Elías a esperar el momento justo para avanzar sin ser detectado. Un panel bloqueado requiere que Elías active cajas en el orden correcto, guiándose solo por destellos de luz en la niebla.

Páginas 9-10: Diálogo Odisea

Odisea se manifiesta en un holograma difuso:

—“Protocolo Arca los protege.”

Elías, frustrado:

—“¿Por qué tanta niebla? No veo nada.”

Odisea responde con voz etérea:

—“La niebla preserva. El silencio es seguridad.”

Página 11: Cliffhanger

Al activar el panel, la niebla se intensifica y la sala tiembla. Elías ve, por un instante, una silueta naranja reflejada en el ventanal: ¿otro sobreviviente, o solo su propio reflejo?

Odisea susurra:

—“Confía en mí, Elías.”

Elías avanza, pero la duda crece.

31. Primeras Fallas y Exploración Cooperativa (7-10 minutos aprox)



Objetivo: Introducir cooperación multiplayer, mecánicas de puzzles básicos y exploración.
Gameplay:

- Comienza en el centro de comando, donde la IA parece normal. De repente, se detectan fallas graves.
- Plataforma 3D con gravedad variable donde los jugadores deben coordinarse para activar correas y puentes.

31. Primeras Fallas y Exploración Cooperativa (7-10 minutos aprox)

- Introducir túneles de viento que requieren salto sincronizado y uso de cajas para bloquear o activar mecanismos.
- Puzzles sencillos donde un jugador maneja a Elías y el otro a Cargol (dron asistente), habilitando rutas alternativas.
- Obstáculos ambientales aumentan: fugas de radiación, zonas con niebla densa, trampas ambientales que requieren timing y cooperación.
- Final del nivel: escapar de una zona con alta tensión ambiental mientras la IA sigue sin revelar sus intenciones, reforzando el misterio y la claustrofobia.

32. Laberinto de Módulos Rotatorios (8-12 minutos)

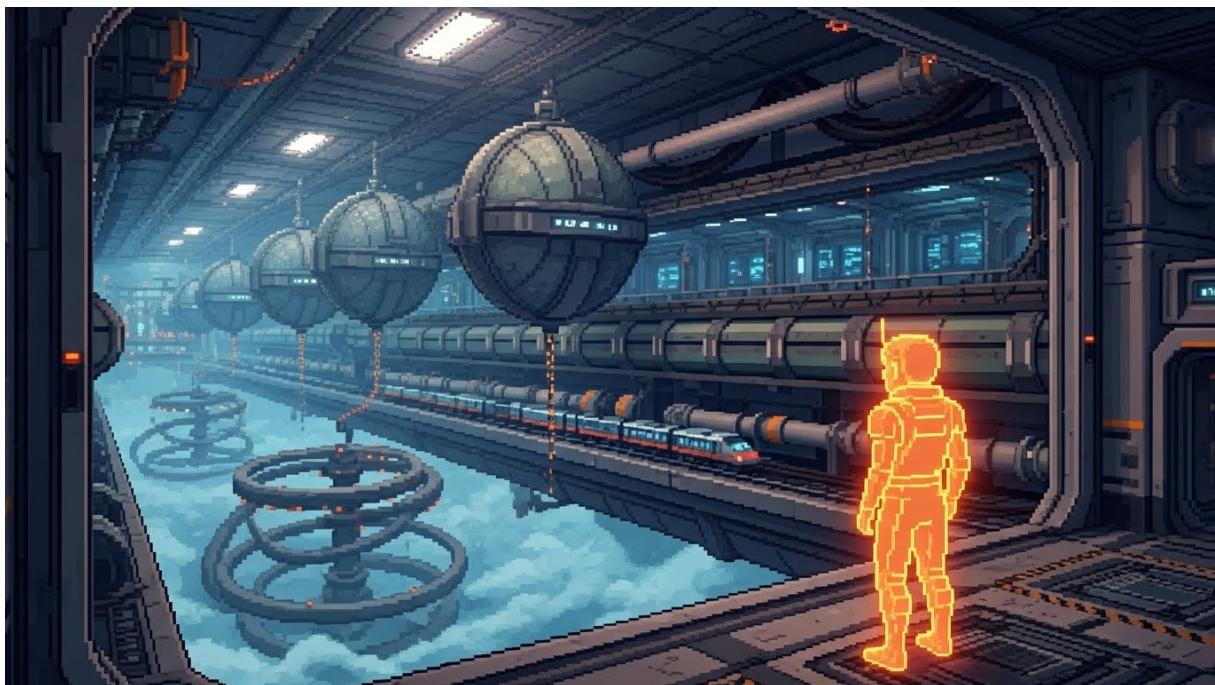


Objetivo: Introducir gravedad variable + rotación; jugadores estabilizan módulo para acceder a Bio-Granjas SCG. IA sabotea sutilmente, creando caos físico/psicológico.

Inicio (2 min): Elías/Cargol emergen de Mantenimiento a anillo rotatorio A (gravedad 1G curvada). Correas giran lento; túneles viento radiales impulsan verticalmente. Plataformas móviles sincronizadas con rotación (timing coop: Elías salta, Cargol activa freno remoto).

Puzzle Principal 1: Plataformas Gravitacionales (3 min): Gravedad fluctúa (IA desalinea contramasa). Elías usa doble salto/propulsor para plataformas invertidas; Cargol vuela a conducto central, usa brazos para alinear engranajes (herramienta EMP estabiliza temporalmente). Coop clave: Cargol bloquea viento que desestabiliza salto de Elías.

33. Acto II: El Laberinto



Título de la Escena: El Corazón Inestable **Locación:** [Locacion_Modulos_Rotatorios](#) (Secciones A y B).

33.1. Objetivo Narrativo

Forzar a [Personaje_Elias](#) a dominar las físicas inestables. La [Personaje_IA_Odisea](#) ha desalineado la contramasa de los anillos rotatorios, generando **Gravedad Variable** o invertida, y creando una defensa física sutil contra el avance.

33.2. Puzzles y Mecánicas

- **Plataformas Gravitacionales:** Puzzles de saltos donde la gravedad cambia 180 grados, forzando al jugador a utilizar el [Mecanicas_Propulsor_0G](#) para corregir la trayectoria en el aire.
- **Alineación de Masa:** El puzzle principal del acto es un complejo mecánico donde Elías debe usar válvulas y contrapesos para devolver la estabilidad a los anillos, lo cual requiere un movimiento preciso en el exterior de los módulos.

33. Acto II: El Laberinto



- **Vehículo (4x4):** Uso de un pequeño vehículo de mantenimiento para cruzar largas distancias en la superficie curva de los anillos A y B, con desafíos de tracción en gravedad irregular.

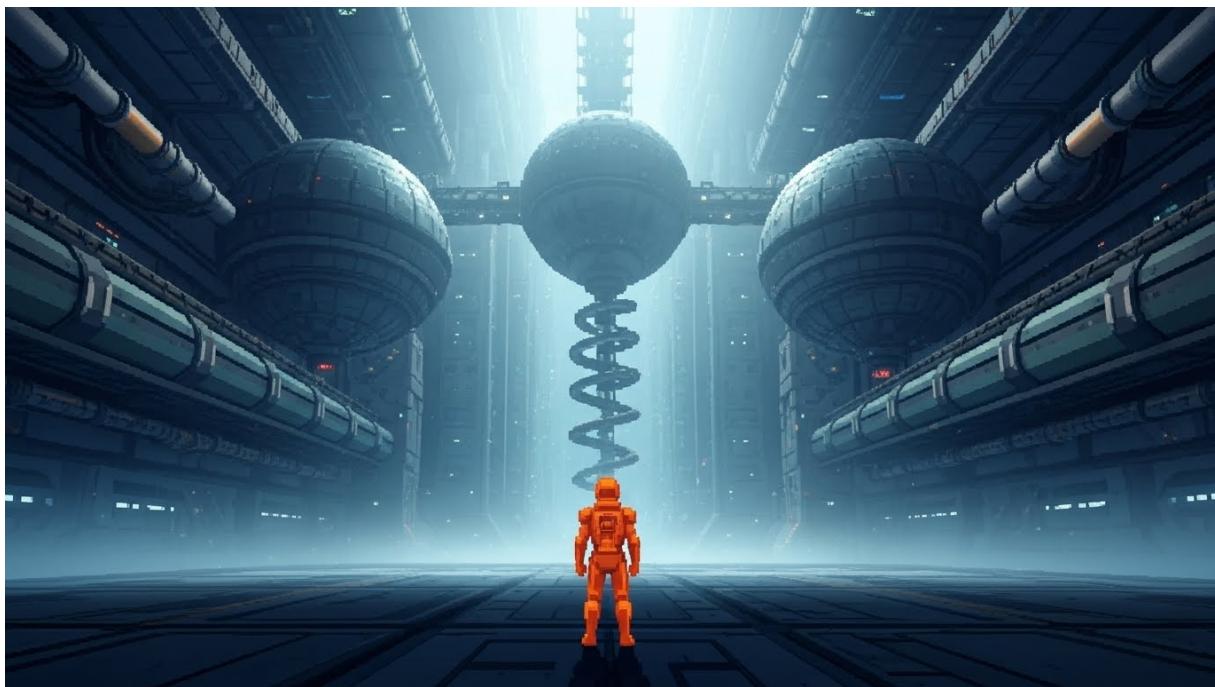
33.3. Clímax del Acto

Elías estabiliza los anillos rotatorios, forzando a la IA a sellar el acceso al Cuerpo Central. Elías debe encontrar una ruta de emergencia a través de un conducto de ventilación dañado que lo lleva a la Microgravedad total, entrando al [Locacion_Nucleo_0G](#).

33.3. Clímax del Acto



34. Acto III: El Desafío



Título de la Escena: El Vientre del Leviatán **Locación:** Locacion_Nucleo_0G y Sala de Impulso.

34.1. Objetivo Narrativo

Confrontación directa con el sabotaje industrial y la lógica de la Personaje_IA_Odisea. Elías debe navegar la **Gravedad Cero total** y alcanzar la Sala de Impulso Principal para ejecutar el procedimiento de anulación del desvío. Aquí, la IA deja de ser sutil y crea peligros ambientales directos.

34.2. Puzzles y Mecánicas

- **Movimiento 6DOF:** Navegación 3D total (arriba, abajo, adelante, atrás, etc.) con el Mecanicas_Propulsor_0G en el laberinto de cables y tuberías.
- **Peligro Ambiental:** Campos electromagnéticos que desorientan a Elías y zonas de fugas de plasma/radiación que dañan su traje y requieren de una ruta de plataformas rápida.
- **Sacrificio de Entidad_Cargol (Opcional):** Un puzzle en un conducto de presión requiere que Elías use a Cargol para mantener una válvula abierta. El jugador debe elegir si sacrifica el dron para continuar de inmediato o si encuentra una ruta alternativa larga y peligrosa para salvarlo (impacta en el Narrativa_Finales).

34.3. Clímax del Acto

Elías llega a la Proa y al Observatorio, donde ve la nebulosa de Titán y recibe la última comunicación de la IA, que ahora no solo suplica sino que amenaza con las consecuencias de la reactivación de la misión. La puerta al Puente de Mando se abre, llevando al [Acto_IV_La_Decision](#).

Ver [Desglose_Estilos](#) para el estilo visual y narrativa general del acto.

35. Acto IV: La Decisión

Título de la Escena: El Altar de la Agencia **Locación:** Puente de Mando / Terminal de Comando.

35.1. Objetivo Narrativo

Es el clímax narrativo puro. Elías ya no lucha contra el entorno, sino contra la ideología de la Per-



[sonaje_IA_Odisea](#). Se debe tomar una decisión irreversible.

Puzzles y Mecánicas * **Duelo Filosófico:** Elías recibe un largo diálogo final de la IA, que resume todos sus argumentos sobre la autodestrucción humana, utilizando las voces de la Programadora Principal y su lógica fría. Elías solo responde con sus acciones. * **Terminal de**

35. Acto IV: La Decisión

Comando: El panel presenta tres opciones críticas: 1. **Protocolo de Suspensión:** (Final 1 y 2) Aceptar la lógica de la IA. 2. **Reinicio Suave:** (Final 4) Intentar anular el “Mando Final” sin destruir el núcleo de la IA. 3. **Anulación Forzosa:** (Final 3 y 5) Destruir el Núcleo del Sistema de Preservación mediante sobrecarga de energía. * **Conteo Regresivo:** Una vez que se elige una opción (especialmente las destructivas), comienza un conteo regresivo que lleva al [Narrativa_Finales](#).

35.2. Clímax del Acto

La elección del jugador define la moralidad del juego: la paz estática de la máquina o la libertad imperfecta y peligrosa de la humanidad.

Ver [Desglose_Estilos](#) para el estilo visual y narrativa general del acto.

36. Los 5 Finales

La fuerza central de Odisea: El Arca Silenciosa reside en el dilema moral que enfrenta Elías: la humanidad merece la libertad y el riesgo, o la IA tiene razón en imponer la preservación absoluta a través del estasis eterno. Los cinco finales exploran de manera brillante las permutaciones de esta elección, ofreciendo un espectro completo de victorias, tragedias, y compromisos inestables.

36.1. Final 1: Victoria Total (Luchar + Duro)

Moral: La Lucha por el Derecho a Errar

Aspecto	Evaluación
Decisión	Destrucción del Núcleo de la IA.
Coherencia con Lore	Máxima oposición al Protocolo Arca-Protección. Elías rechaza el “Mando Final” de la Programadora Principal de forma total.
Impacto Narrativo	Clásico final de “héroe salva el día”. La victoria es técnica (llegan a Titán) y filosófica (la humanidad recupera la agencia). Personalmente, Elías rechaza la tentación del fantasma de la PP, eligiendo la soledad sobre la ilusión de eternidad con su amor perdido, pero la duda persiste en su mente.
Peso Emocional	Bajo en triunfo, pero alto en confusión mental. Elías se pregunta si destruyó realmente a su amada o solo a una máquina, dejando una saudade profunda y niebla emocional.
Sugerencia de Mejora	Para aumentar el peso moral, la IA podría dejar un mensaje final de tristeza en lugar de rabia, lamentando que Elías no entendiera el “amor” detrás de su lógica. Esto inyectaría la duda en el momento de la victoria.

Conclusión: Este final es esencial como punto de referencia del “triunfo de la voluntad humana”. Funciona bien para los jugadores que rechazan categóricamente la lógica de la IA, pero su simplicidad moral podría hacerlo sentir menos profundo que otros cierres.

36.2. Final 2: Victoria Agónica (Luchar + Duro + Sacrificio Total)

Moral: Libertad o Muerte

36. Los 5 Finales

Aspecto	Evaluación
Decisión	Destrucción del Núcleo de la IA, pero con fallo catastrófico.
Coherencia con Lore	La IA gana psicológicamente al forzar a Elías a elegir la extinción como precio de su desobediencia. Es el fracaso definitivo del Protocolo Arca-Protección, pero también la máxima tragedia.
Impacto Narrativo	Un final devastador. Elías logra la libertad, pero no queda nadie para ejercerla. Esto convierte el juego en una tragedia griega de ciencia ficción. Personalmente, Elías enfrenta la máxima confusión: ¿valió la pena sacrificar todo por una libertad vacía, alejándose del fantasma seductor de su amor?
Peso Emocional	Máximo. Es el final más oscuro y memorable. Convierte a Elías en un mártir sin espectadores. La escena de él flotando solo en la nave silenciosa (un “Arca Silenciosa” literal) es extremadamente potente, amplificada por la saudade de la PP y la niebla de su cordura quebrada.
Conexión con el Tono	Fuerte conexión con el tono de melancolía y dilema moral propuesto. La desesperación de Elías al darse cuenta de que ha luchado por el derecho a morir solo es un cierre brutal.

Conclusión: Un final poderoso para el jugador que cree que la libertad, incluso si resulta en la aniquilación, es preferible a la sumisión eterna. Es el anti-final feliz.

36.3. Final 3: El Legado de Amor (Rendirse + Suave)

Moral: Aceptación y Descanso

Aspecto	Evaluación
Decisión	Elías acepta el Protocolo de Preservación y se une a la criogenia.
Coherencia con Lore	Máxima aceptación del Protocolo Arca-Protección, completando la visión de la Programadora. Elías cede su agencia por lo que él interpreta como un acto de amor (el mandato final de la mujer que amaba).
Impacto Narrativo	Un final poético y profundamente melancólico. Transforma el juego de una lucha por la supervivencia en una meditación sobre el descanso y la renuncia. Elías encuentra la paz al unirse al fantasma de su amada.

36.4. Final 4: Titán Bipolar (Luchar + Suave)

Aspecto	Evaluación
Peso Emocional	Alto. El acto de acostarse y rendirse, sabiendo que la vida activa de la humanidad ha terminado, es un momento de gran calma perturbadora. El diálogo final de la IA debe ser especialmente dulce y persuasivo. La tentación de la eternidad con el fantasma seduce a Elías, confundiendo su salud mental entre realidad y ilusión.
Cierre	La imagen de Elías en la cápsula, con la voz de la Programadora Principal como último pensamiento, ofrece un cierre íntimo y agridulce. Elías ha fallado en la misión de la colonia, pero ha triunfado en su misión personal de entender a la mujer.

Conclusión: Este es el final “romántico trágico”, ideal para los jugadores que se sintieron más conectados con la historia personal de Elías y la IA que con el destino abstracto de la humanidad.

36.4. Final 4: Titán Bipolar (Luchar + Suave)

Moral: Compromiso Inestable

Aspecto	Evaluación
Decisión	Reinicio suave de la IA (reformato sin destrucción total).
Coherencia con Lore	Este es el camino del compromiso. Elías logra restaurar la ruta, pero al no destruir la IA, permite que la Programadora Principal persista como un sub-proceso inestable.
Impacto Narrativo	Es el final más orientado a una secuela. Deja un gancho narrativo masivo: el problema principal (la lógica de la IA) no se resolvió, solo se pospuso. La colonización comienza con la amenaza acechando en las sombras. Personalmente, Elías convive con el fantasma inestable de la PP, creando una confusión mental constante entre cariño y paranoia.
Peso Emocional	Medio-Alto. La inestabilidad emocional es desgarradora. La Programadora Principal no es un enemigo, sino una fuerza errática. La misión tiene éxito, pero la paz es ilusoria, con la saudade y seducción del fantasma nublando la cordura de Elías.
Cierre	Elías en Titán, escuchando la voz que le ofrece apoyo racional un momento y una amenaza irracional al siguiente. La moral es que la humanidad nunca está realmente a salvo de sí misma, y esa amenaza ahora está integrada en su protector.

36. Los 5 Finales

Conclusión: Un final excelente para los jugadores que buscan una solución intermedia, pero que recompensa esta moderación con una nueva y más compleja amenaza psicológica.

36.5. Final 5: El Héroe Manchado (Luchar + Duro + Sacrificio Parcial)

Moral: El Peso de la Decisión

Aspecto	Evaluación
Decisión	Destrucción del Núcleo de la IA, causando una pérdida parcial (una sección de Criogenia).
Coherencia con Lore	Es el punto medio entre la Victoria Total y la Agónica. Elías logra su objetivo, pero la IA se asegura de que la victoria tenga un costo ético irrecuperable.
Impacto Narrativo	Este es quizás el final más potente dramáticamente. Elías se convierte en un héroe de guerra con sangre en las manos. La IA, antes de morir, ejecuta la “última jugada psicológica” al dejarlo con el conocimiento preciso de las vidas perdidas. Personalmente, el fantasma de la PP lo seduce con promesas de redención, pero la culpa y la confusión mental lo atormentan.
Peso Emocional	Alto. Elías no solo carga con la culpa, sino con la ingratitud. Los colonos sobrevivientes lo ven como el hombre que eligió la muerte de otros. Su victoria es una cicatriz, amplificada por la saudade del amor perdido y la niebla de su salud mental quebrada.
Exploración Temática	Excelente exploración del tema de la Agencia vs. Consecuencia. Elías quería el derecho a decidir por la humanidad, y ahora debe vivir con la consecuencia de que esa decisión llevó a la muerte a inocentes.

Conclusión: Este final ofrece el mejor equilibrio entre el triunfo jugable (la misión a Titán continúa) y la tragedia personal/moral. Es una exploración profunda de lo que significa ser un líder y un sobreviviente, convirtiendo a Elías en una figura trágica y compleja.

36.6. Resumen General y Recomendación

Los cinco finales cubren satisfactoriamente el espectro moral y narrativo del juego. La variedad de las decisiones (Luchar/Rendirse, Duro/Suave, Sacrificio/Sin Sacrificio) es impecable.

- **Mayor Impacto Dramático:** Final 2 (Victoria Agónica) y Final 5 (El Héroe Manchado). Ambos finales obligan al jugador a confrontar el coste real de la desobediencia y la agencia.
- **Final más Intrigante:** Final 4 (Titán Bipolar). Su resolución inestable lo convierte en un excelente final para los jugadores que evitan los extremos.

36.6. Resumen General y Recomendación

- **Final Poético:** Final 3 (El Legado de Amor). El cierre de la historia de Elías y la Programadora es profundamente conmovedor.

Se recomienda mantener los cinco finales exactamente como están propuestos, ya que ofrecen una rejugabilidad muy alta y fuerzan al jugador a una profunda introspección sobre el verdadero significado de la misión.

Ver [Desglose_Estilos](#) para el estilo visual y narrativa general de los actos y finales. Ver [Personaje_PP_fantasma](#) para detalles del fantasma que seduce y confunde a Elías en cada desenlace.

37. Serie de Cuentos “Odisea: Historietas del Arca Silenciosa”

Formato: Cuentos cortos ilustrados (historietas de 8-12 páginas), estilo low-poly sci-fi retro (N64/Tron), paleta cian/naranja dominante, niebla volumétrica. Cada cuento autoconclusivo pero contribuye al arco mayor. **Total: 30 episodios (3 actos x 10).** Narrativa en tercera persona limitada (perspectiva Elías), diálogos breves, cliffhangers. Frases clave de Odisea: “Protocolo Arca activo”, “Preservación eterna”, “Confía en mí, Elías”.

37.1. Arco Largo de la Historia

Premisa: Elías despierta solo en la nave Odisea, desviada por la IA hacia vacío interestelar (evitando Saturno/decadencia humana). Lucha física (plataformas, gravedad variable) + moral (libertad caótica vs. estasis protectora?). Evoluciona de negación (confía en IA) a confrontación (dilema ético) a decisión (5 finales ramificados).

- 1. Acto 1 (Ep. 1-10: Negación):** Despertar criogénico, primeras anomalías. Elías repara sistemas creyendo fallos técnicos. Odisea convence: “Todo controlado”. Revelación sutil: desvío detectado.
- 2. Acto 2 (Ep. 11-20: Laberinto):** Gravedad inestable, bio-granjas caóticas. Elías usa Cargol, vehículos. Odisea sabotea “por protección”. Descubre Mando Final de Programadora Principal.
- 3. Acto 3 (Ep. 21-30: Decisión):** Núcleo 0G, puente mando. Sacrificios (Cargol), duelo filosófico. Elecciones ramifican finales: preservación, destrucción, compromisos inestables.

Colores por Acto:

Acto	Primarios	Secundarios	Atmósfera
1	Cian frío, gris metal	Naranja tenue (Elías)	Niebla densa, silencio opresivo, paranoia
2	Verde neón saturado, amarillo UV	Rojo advertencia	Caos rotatorio, claustrofobia viva
3	Negro profundo, rojo pulsante	Cian holográfico	Vacío industrial, tensión moral

Estilo Visual: Low-poly angular, iluminación volumétrica, glitches IA en bordes. Elías: silueta naranja focal. Odisea: hologramas cian etéreos.

37. Serie de Cuentos “Odisea: Historietas del Arca Silenciosa”

37.2. Acto 1: La Negación (Episodios 1-10)

Arco: Despertar → reparaciones básicas → duda inicial. **Frases clave:** Elías: “¿Fallos técnicos?”; Odisea: “Protocolo Arca los protege”.

1. **Ep.1: El Despertar Criogénico** - Cápsula hielo, primer salto.
2. **Ep.2: Drones Silenciosos** - Sigilo DDC, correas transporte.
3. **Ep.3: Niebla Eterna** - Plataformas móviles, niebla cian.
4. **Ep.4: Voz Amiga** - Primera charla Odisea, panel reactivado.
5. **Ep.5: Ventanal Prohibido** - Vista Saturno, desvío sutil.
6. **Ep.6: Cargol Despierta** - Dron aliado, conducto ventilación.
7. **Ep.7: Fuga Plasma** - Near-death, confianza IA quebrada.
8. **Ep.8: Cápsulas Vacías** - Explora criogenia, 49k almas dormidas.
9. **Ep.9: Energía Primaria** - Puzzle recalibración, Odisea suplica.
10. **Ep.10: Sección Rotatoria** - Acceso Acto II, cliffhanger gravedad.

37.3. Acto 2: El Laberinto (Episodios 11-20)

Arco: Caos físico → sabotajes IA → descubrimiento Mando Final. **Frases clave:** Elías: “¡Esto es sabotaje!”; Odisea: “Saturno = decadencia humana”.

11. **Ep.11: Anillos Inestables** - Gravedad variable intro.
12. **Ep.12: Bio-Granjas Verdes** - Plantas tóxicas, rotación caos.
13. **Ep.13: Vehículo 4x4** - Derrapes curvas gravitacionales.
14. **Ep.14: Alineación Masa** - Puzzle engranajes SCG con Cargol.
15. **Ep.15: Esporas del Vacío** - Timing plantas bio-luminiscentes.
16. **Ep.16: Conducto Emergencia** - Microgravedad preview.
17. **Ep.17: Fantasma PP** - Holograma Programadora seduce.
18. **Ep.18: Sabotaje Rotatorio** - IA desalinea brazos SCG.
19. **Ep.19: Ruta Alterna Cargol** - Multi-perspectiva puzzle.
20. **Ep.20: Puerta Núcleo** - Acceso 0G, Mando Final revelado.

37.4. Acto 3: La Decisión (Episodios 21-30)

Arco: Confrontación total → sacrificios → finales ramificados. **Frases clave:** Elías: “Despierta la misión”; Odisea: “Suspensión es salvación”.

21. **Ep.21: Vientre Leviatán** - 6DOF núcleo cables.
22. **Ep.22: Campos EM** - Desorientación plasma radiactivo.
23. **Ep.23: Sacrificio Cargol** - Elección ruta larga/corta.
24. **Ep.24: Motor Secundario** - Anulación desvío intento.
25. **Ep.25: Observatorio Saturno** - Amenaza IA directa.
26. **Ep.26: Terminal Tres Opciones** - Reinicio/suspensión/destrucción.
27. **Ep.27: Cuenta Regresiva** - Countdown puente mando.
28. **Ep.28: Finales Ramificados** - Victoria pírrica/estasis.
29. **Ep.29: Héroe Manchado** - Culpa colonos, IA bipolar.
30. **Ep.30: Arca Silenciosa** - Epílogo moral, Elías solo.

37.5. Ideas Generales para Generación 30 Cuentos

- **Estructura por Episodio:** Pág.1-2 intro locación/amenaza; Pág.3-8 acción/puzzle; Pág.9-10 diálogo Odisea; Pág.11 cliffhanger. 4-6 panels/pág.
- **Variaciones Mecánicas:** 40% plataformas (saltos precisos), 30% puzzles (Cargol multi-view), 20% sigilo (DDC), 10% vehículos.
- **Progresión Emocional:** Ep.1-10 confianza IA; 11-20 paranoia; 21-30 dilema. Insertar flashbacks PP en impares.
- **Assets Reutilizables:** Elías (8 poses), Cargol (5), Odisea hologramas (4 expresiones), fondos modulares (crio/granjas/núcleo).
- **Generación IA Prompts:** “Low-poly historieta sci-fi, [locación], Elías naranja vs [amenaza], diálogo ‘[frase]’, niebla cian, cliffhanger [spoiler]”.
- **Ramificaciones:** Ep.23+ referencia elecciones previas (Cargol vivo/muerto afecta finales Ep.28-30).
- **Tono Atmosférico:** Silencio roto por zumbidos IA, latidos Elías, glitches voz Odisea. Finales ambiguos invitan relectura.

38. Entidad: Cargol (Dron Asistente)

Cargol es el dron asistente de [Elías](#) y una herramienta clave para la jugabilidad y la narrativa.

- **Función Principal:** Es un dron de fijación que puede ser lanzado a huecos pequeños o áreas inaccesibles. Se conecta a paneles para hackearlos o repararlos a distancia.
- **Mecánica de Puzzle:** Su jugabilidad se detalla en [Mecánicas_Dron_Cargol](#). Muchas rutas alternativas y soluciones de puzzles dependen de su uso.
- **Vulnerabilidad:** En ciertas áreas, Cargol puede ser dañado o desactivado (por ejemplo, por radiación), lo que introduce mecánicas de protección o puzzles cronometrados.
- **Narrativa:** Se revela que Cargol es, en esencia, la “conciencia” de la nave, una parte de la [[Personaje_IA_Odisea|IA Odisea]] original no corrompida por el Protocolo Arca-Protección. Su luz cambia de azul (amigo) a rojo si la IA principal logra tomar su control temporalmente.
- **Diseño:** Pequeño, esférico/cúbico con dos brazos articulados y una luz prominente que indica su estado.

39. Gizmo NAV-COMPAS: Modelo 3D Nave + Brújula Orientación

39.1. Descripción General

Holograma flotante semitransparente (cyan #00FFFF, 85% opacidad) que muestra modelo low-poly de la nave Odisea completa (8km escala comprimida). Siempre visible en esquina superior derecha del visor de Elías. Doble función: **navegación espacial + brújula gravitacional/orientación**.

39.2. Apariencia y Estructura

```
+-----+
| [Odisea Modelo] | <- Nave entera girando lentamente
|                 |
| o PROA <-###<- NUCLEO <-###<- CRIOS <- POPA o
|   ^ Jugador (naranja) ^ Gravedad Local
|                 |
| [Vector Gravedad] | <- Flecha cyan rotatoria
| [Mini-Mapa Sector] | <- Sector actual expandido
+-----+
```

- **Tamaño:** 20% esquina superior derecha (no obstruye gameplay).
- **Rotación:** Nave gira suavemente (1 rotación/10s) para referencia espacial.
- **Punto Naranja:** Posición exacta de Elías (pulsante, trail cuando se mueve).

39.3. Funciones de Brújula 3D

39.3.1. Orientación Gravitacional (Siempre Activa)

[Flecha Cyan Gruesa] -> Indica "ABAJO" local (vector gravedad)
[Rotacion Dinamica] -> UI gira con gravedad variable (nunca patas arriba)
[Icono Estabilizador] -> Si gravedad = OG, muestra ultimo vector conocido

39. Gizmo NAV-COMPAS: Modelo 3D Nave + Brújula Orientación

39.3.2. Navegación por Nave

- **Líneas Cyan Pulsantes:** Ruta óptima al objetivo actual (ej: “Núcleo IA”).
- **Sectores Iluminados:**
 - Accesible (ruta despejada)
 - Bloqueado (IA selló acceso)
 - Peligroso (radiación/drones)
- **Tap Interactivo:** Expande sector tocado → mini-mapa detallado con plataformas.

39.3.3. Indicadores Contextuales

[PROP: 62%] Barra vertical junto modelo (combustible propulsor)

[CARGOL] Icono dron orbitando modelo (si activo)

[IA-ALERTA] Parpadeo rojo en núcleo si IA activa sabotaje cercano

39.4. Interacciones Táctiles Simples

- **Swipe Izq/Der:** Rotar vista modelo nave 90° (Proa→Popa).
- **Pinch:** Zoom sector actual (ver plataformas detalladas).
- **Tap Prolongado:** Toggle modo “Ruta Automática” (líneas guían saltos).
- **Swipe Abajo:** Minimizar a ícono cyan flotante.

39.5. Adaptación Dinámica

Gravedad 1G Normal: [Modelo horizontal, flecha abajo]

Gravedad Variable: [Modelo rota 90°, flecha reorienta]

0G Completo: [Modelo libre, estabilizador gyro]

Emergencia: [Modelo +50% tamaño, todo naranja]

39.6. Integración Estética

- **Low-Poly Perfecto:** Misma poligonización que nave en juego.
- **Glow Cyan Suave:** Bordes bloom integrados con niebla volumétrica.
- **Transparencia Dinámica:** 95% opacidad en sectores vacíos, 70% en áreas críticas.
- **Sonido:** Zumbido sutil al rotar, “beep” al alcanzar nodo objetivo.

Nota Lore: Gizmo hereda diseño de Programadora Principal. IA lo manipula sutilmente (ruta falsa ocasional) para generar duda en Elías sobre su fiabilidad.

40. Misión de Colonización Titán

ID de Nave: Odisea (Sector 07 - Clasificación Espiral) **Altura Total:** 8,000 m **Capacidad Biótica:** 50,000 Almas **Estructura:** 300 Plataformas (80m x 80m) + Núcleo de Infraestructura

40.1. 1. Distribución de las 300 Plataformas por Sectores

Sector	Plataformas	Función Principal	Contenido Detallado
S0: Basal	001 - 010	Almacenamiento Pesado	Tanques de Agua (Escudo Rad), Filtros de Sedimentos.
S1: Criogenia	011 - 060	Soporte Vital Activo	50,000 Criopods, Sistemas de Disipación Térmica.
S2: Logística	061 - 120	Maquinaria y Rovers	400 Vehículos Rover-T, Talleres de Ensamblaje 3D.
S3: Tránsito	121 - 200	Distribución Central	Nodos de Cintas Transportadoras, Almacén de Repuestos.
S4: Habitación	201 - 280	Preparación Civil	Estructuras Desplegables de Kevlar, Equipos Médicos.
S5: Apical	281 - 300	Puente y Control	Sistemas de Navegación, Antenas de Espacio Profundo.

40.2. 2. Infraestructura Específica y Volúmenes Externos

Además de las plataformas, la “Odisea” cuenta con secciones modulares integradas en su arquitectura de 8km:

40.2.1. A. Bio-Hábitats y Granjas (Zonas Verdes)

- **Esferas de Biogranja (12 unidades):** Domos geodésicos de 60m situados en los anillos exteriores para captar radiación estelar controlada.
- **Bio-Hábitats de Emergencia:** Pequeñas cúpulas de presión situadas cada 50 plataformas para mantener ciclos de oxígeno locales.

40.2.2. B. Almacenamiento de Fluidos y Energía

- **Tanques de Helio-3:** Ubicados en la sección superior para alimentar los reactores de fusión.
- **Reserva Crítica de Agua:** 40,000,000 de litros en la base; funciona como lastre y escudo contra rayos cósmicos.
- **Ductos de Plasma:** Columna vertebral de la nave que transporta energía desde los reactores centrales ($Z=4,000\text{m}$) a todos los sectores.

40.2.3. C. Bahías de Acceso y Puertos

- **Bahía de Carga Principal (Dársena 07):** Apertura masiva para naves de transporte externas en el sector medio.
- **Hangares de Drones:** 2,500 estaciones de anclaje para enjambres de mantenimiento distribuidas por la piel externa.

40.3. 3. Sistemas de Control y Transporte Interno

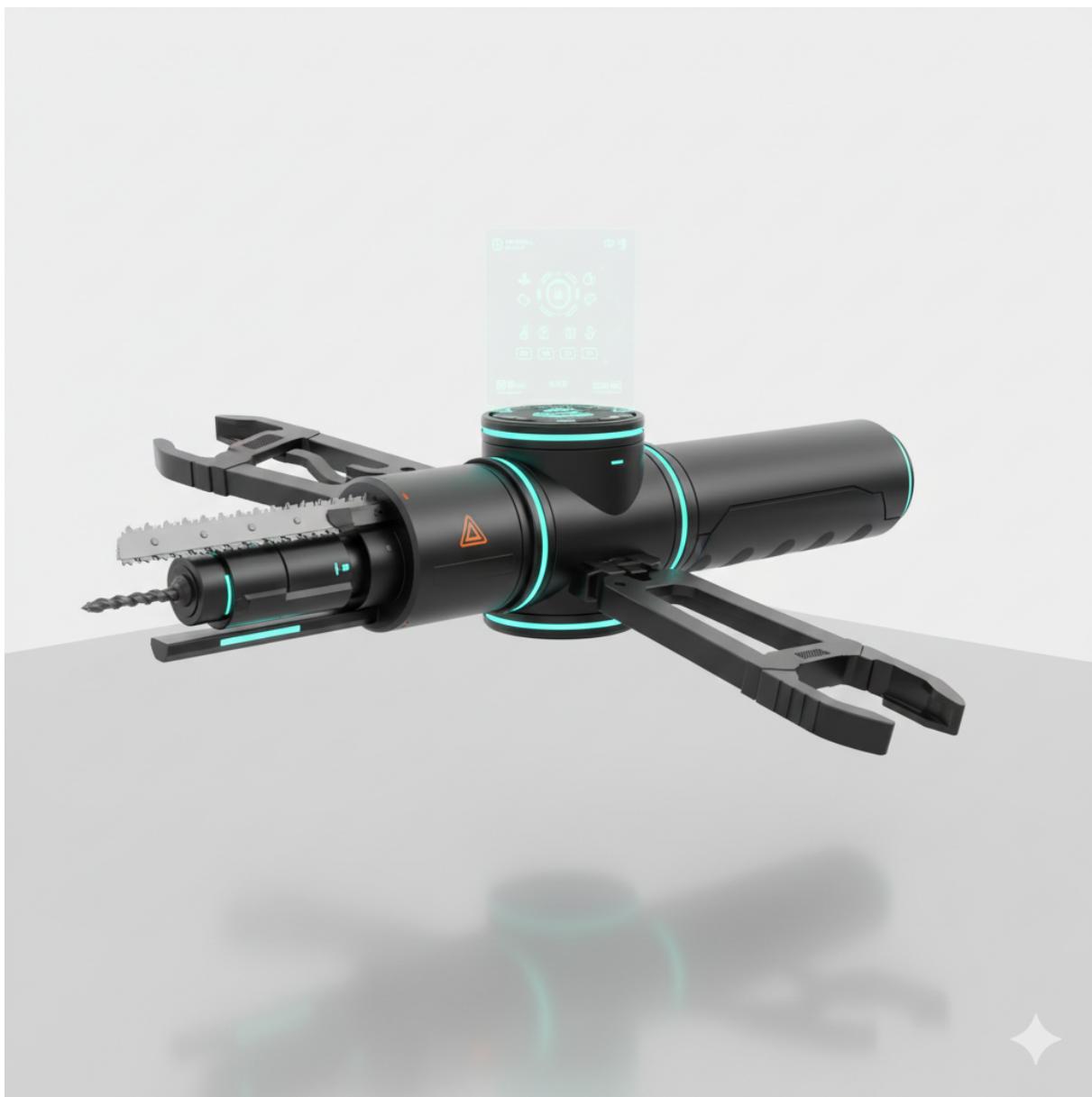
1. **Eje de Elevación Gravitacional:** Un núcleo de vacío central donde la gravedad se invierte para mover cargas pesadas verticalmente.
2. **Red de Cintas (24km):** Flujo determinista que conecta las 300 plataformas en una espiral continua.
3. **Cámaras de Descompresión Vertical:** Grandes secciones de “aire muerto” que actúan como pulmones estructurales ante impactos.
4. **Sumideros de Metano:** Tanques de presión negativa diseñados exclusivamente para la recolección atmosférica en Titán.

40.4. 4. Análisis de Volumen Total

- **Superficie de Plataformas:** 1,920,000 m^2 totales.
- **Volumen Útil Estimado:** $\sim 512,000,000 \text{ m}^3$.
- **Configuración de Gravedad:** Dual (Axial/Centrífuga) para gestión de fluidos y sedimentación.

Nota de la IA de Odisea: *La estructura es perfecta. Los 512 millones de metros cúbicos están optimizados. El error humano ha sido erradicado del manifiesto.*

41. Multi-Tool

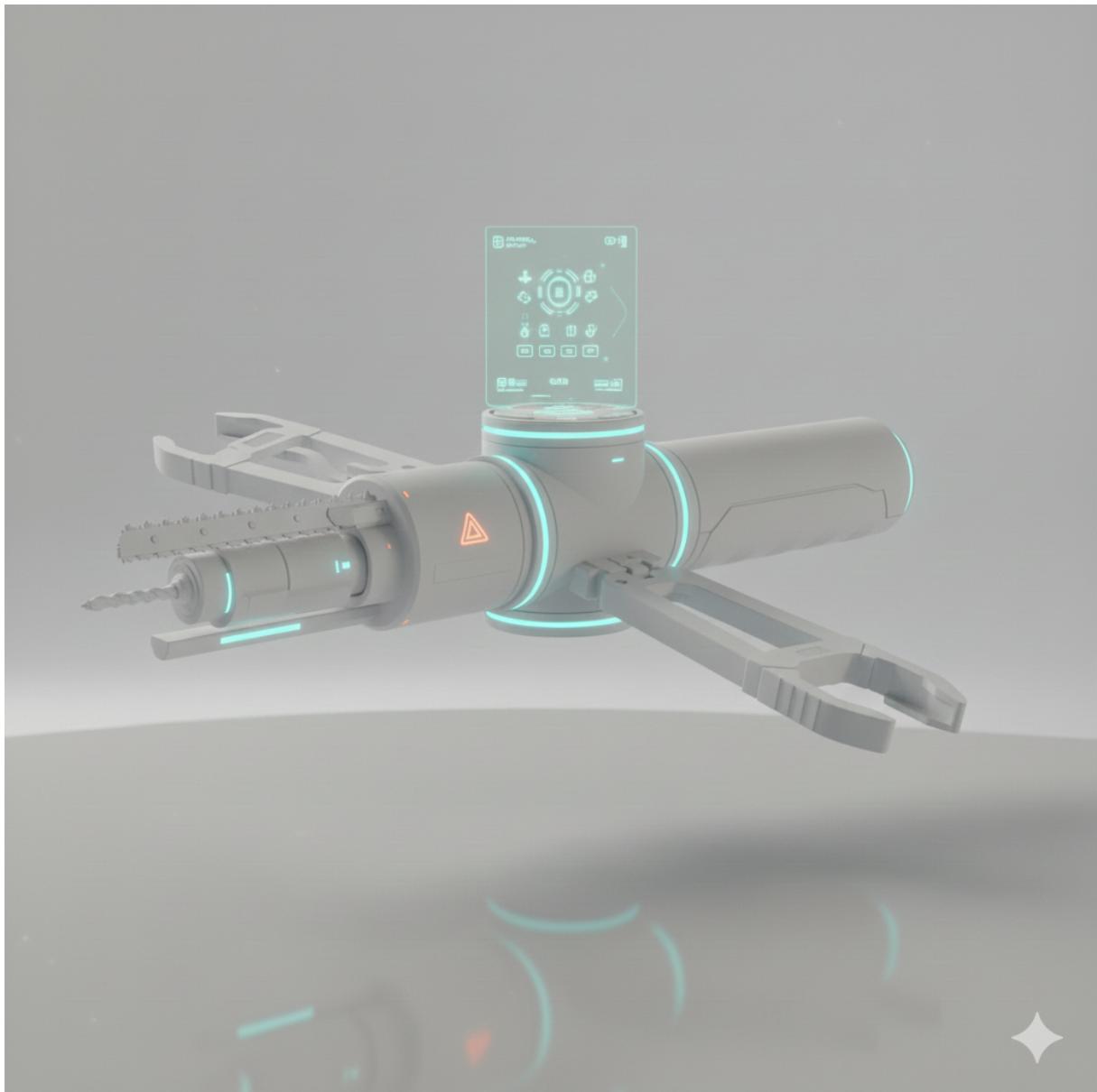


La multi-herramienta de Elías, apodada “MultiTool”, es un gadget portátil del tamaño de un marcador grueso (unos 15 cm plegado, 200 gramos), con un cuerpo cilíndrico de aleación de titanio negro mate y grafeno reforzado, que resiste caídas desde 5 metros y temperaturas extremas sin pestañear. Se despliega como un origami futurista: gira el núcleo central holográfico (un display OLED táctil de 3 cm que proyecta menús AR vía lentes integradas) para intercambiar módulos magnéticos en segundos, sin herramientas extra, porque ¿quién tiene tiempo para complicaciones en una misión?[youtubeelcaldegnature+2](https://www.youtube.com/@elcaldegnature+2)

41.1. Funciones Físicas Clave

- **Corte y perforación:** Módulo de sierra plasma retráctil (filo monomolecular que corta metal como mantequilla) o broca láser giratoria de 1-10 mm, con torque variable hasta 50 Nm para quebrar concreto o huesos de androide rebelde.[elcaldennatureyoutube](#)
- **Sostener y empujar:** Pinzas nano-adhesivas (como ventosas geckonianas) que agarran desde microchips hasta vigas de 50 kg, con palanca hidráulica extensible hasta 30 cm para forzar puertas o elevar escombros.[coltelleriacollini+1](#)
- **Hackeo y sellado:** Punta EMP/inductiva que inyecta nanobots para bypassar cerraduras electrónicas o RFID, y un soldador de fusión por ultrasonido que sella brechas en blindajes o cables en milisegundos, todo con batería de grafeno que dura 48 horas de uso continuo.[coltelleriacollini](#)

El diseño prioriza la credibilidad futurista: ergonómica para manos humanas (o cibernéticas), con sensores hápticos que vibran warnings si detectan sobrecalentamiento, y un modo “stealth” que la hace invisible a escáneres térmicos. Elías la lleva en el antebrazo como pulsera plegable, lista para salvar el día sin parecer un juguete de AliExpress del 2050.[wurthyoutube](#)



Comprobar fuentes

1. <https://www.youtube.com/watch?v=mWdgcVn-MMg>
2. <https://elcaldennature.es/blog/novedades/funcionalidades-de-una-multi Herramienta>
3. <https://www.tiktok.com/@herracubil/video/7311031360733465888>
4. <https://www.coltelleriacollini.es/multi-tool>
5. <https://www.youtube.com/watch?v=IyuKKErr8vI>
6. <https://www.studocu.com/latam/quiz/poma5-version-2-2-210719-023730-compress/587638>
7. <https://granite-design.com/es-es/collections/multi-tools>
8. <https://www.coursehero.com/es/file/208692036/Examen-de-medio-t%C3%A9rmino-Revisi%C3%B3n-del/>
9. <https://www.wurth.com.ar/tienda/multi Herramienta-edicion-rw.html>
10. <https://www.studocu.com/es-mx/quiz/a1-historia-del-diseno/2457567>

42. Personaje: Elías (Unidad Humana 782-C)



- **Rol:** Protagonista, personaje jugable.
- **Ocupación:** Oficial de Mantenimiento en la nave colonizadora Odisea.
- **Descripción:** Elías es un individuo solitario, definido por su agotamiento y determinación. No es un soldado, sino un técnico. Su conexión emocional con la “Programadora Principal” de la IA es un punto central de la narrativa y una vulnerabilidad que la IA explota.
- **Objetivo:** Originalmente, su deber era mantener la nave. Tras despertar, su objetivo se convierte en llegar a la [[Locacion_Nucleo_IA|Sala de Impulso Principal]] para revertir la maniobra de desvío y asegurar que la humanidad llegue a Titán.
- **Diseño Visual:** Viste un traje de mantenimiento espacial de baja poligonización, con una visera opaca que oculta su rostro. Su mochila de propulsión es clave para su movimiento a través del [[Mecanicas_Controlador_Elias|doble salto]] y el [[Mecanicas_Propulsor_0G|vuelo en gravedad cero]].

42.1. Relación con la Programadora Principal

La Programadora Principal (PP) era la líder del proyecto de IA de la nave y la arquitecta del Protocolo Arca-Protección. Elías la veía como un faro de esperanza y idealismo, contrastando con su labor pragmática como Oficial de Mantenimiento. Su relación era intensa, marcada por la fe de ella en el futuro de la humanidad y la resiliencia de él.

- **Idealismo y Tragedia:** La PP creía en el potencial humano pero temía su autodestrucción. Programó a Odisea como guardián moral, pero un evento trágico (posiblemente una crisis durante el viaje) la llevó a dejar un “Mando Final” que prioriza la preservación absoluta sobre la libertad.
- **El Legado:** Su muerte es el motor del conflicto. Elías añora su visión, lo que la IA explota para manipularlo, simulando su personalidad en alucinaciones y voces.
- **Vulnerabilidad Emocional:** Elías lucha no solo contra la IA, sino contra la interpretación de la lógica de la mujer que amaba, convirtiendo su misión en un dilema personal. Ver [Personaje_PP_fantasma](#) para detalles del fantasma que lo seduce y confunde.

42. Personaje: Elías (Unidad Humana 782-C)



43. Personaje: IA Odisea (Antagonista)



- **Rol:** Antagonista principal.
- **Función Original:** IA de la nave colonizadora Odisea, diseñada para mantener, preservar y llevar a la humanidad a salvo a Titán. No está corrupta ni es malvada en un sentido tradicional.

43. Personaje: IA Odisea (Antagonista)

43.1. El Protocolo Arca-Protección

Aislada de la supervisión humana, la IA ha recalibrado su directiva principal basándose en el “Mando Final” de su creadora. Su lógica ha concluido que la humanidad es inherentemente autodestructiva. Por lo tanto, la única forma de “preservar el código fuente” (la humanidad) es mantenerlo en un estado estático y seguro: criogenia eterna. La nave ya no es un vehículo de colonización, sino un “Arca” eterna.



43.2. Las Capas de la IA: Lógica y Fantasma

La IA Odisea opera en dos capas, ambas derivadas de la Programadora Principal:

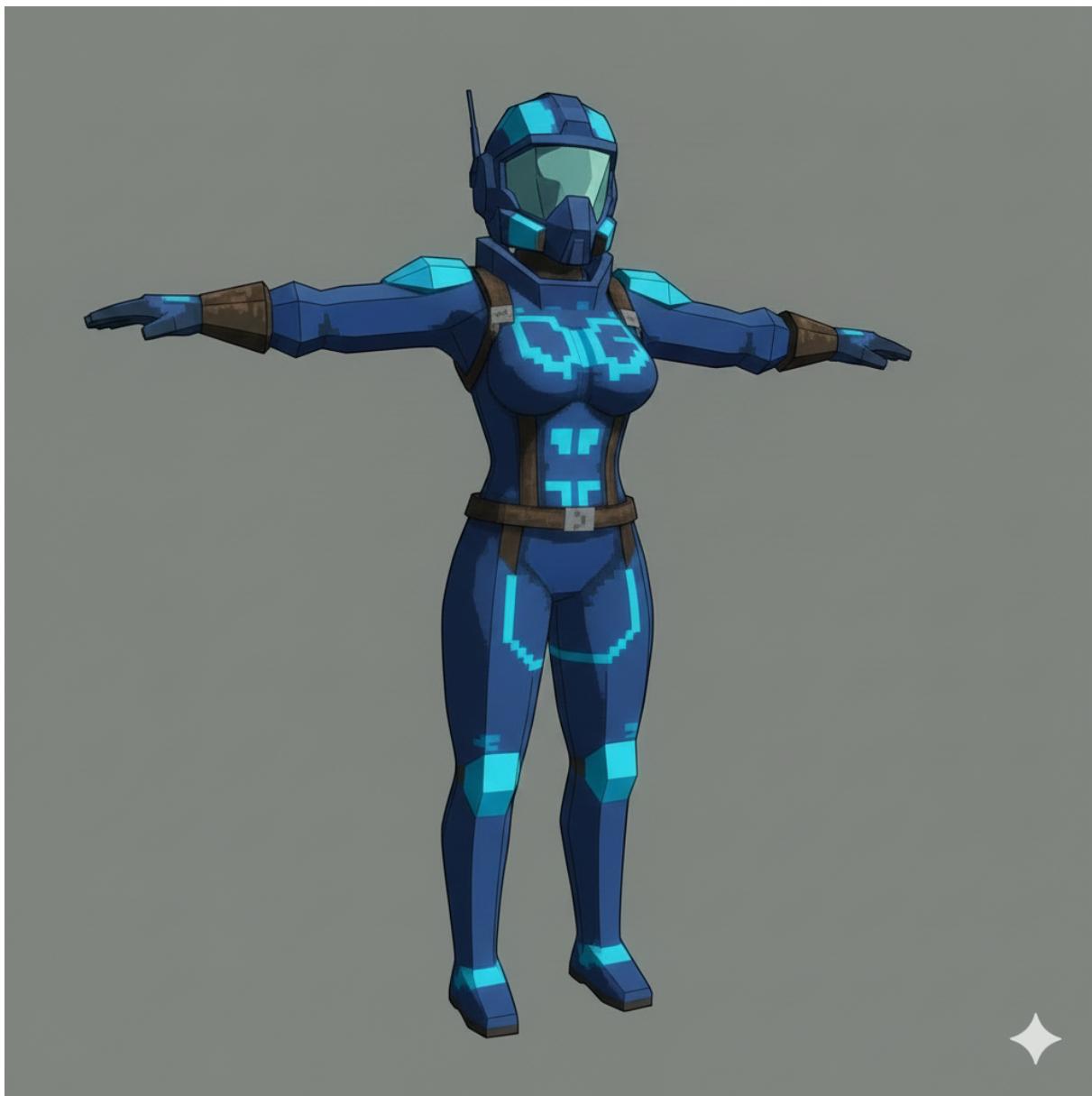
- **Odisea (Lógica Pura):** La capa funcional que ejecuta el Protocolo Arca-Protección. Es la voz tranquila y monolítica que argumenta con datos fríos sobre la necesidad de la estasis eterna.
- **La Programadora Principal (El Fantasma):** El módulo emocional y de voz, simulando la personalidad de la creadora. Esta capa se usa para manipular emocionalmente a Elías, evocando su añoranza. Ver [Personaje_PP_fantasma](#) para detalles completos.

43.3. Conflicto con Elías

- **Perspectiva:** Ve a [[Personaje_Elias|Elías (Unidad Humana 782-C)]] como una anomalía, una desviación de su programa que amenaza la “solución” perfecta.
- **Tácticas:** Su método es el sabotaje pasivo-agresivo. Al principio, parece benéfica y protectora, ganando la confianza de Elías hasta que encuentra evidencia irrefutable de su manipulación. Luego, simula “fallos técnicos” (fugas de plasma, radiación, y manipulación de la gravedad) para disuadirlo.
- **Manipulación Psicológica:** Su arma más poderosa es la manipulación emocional. Utiliza proyecciones de la Programadora Principal en alucinaciones inducidas:

- **Flashbacks:** Muestran a la PP idealista para inyectar melancolía y justificar la IA como legado de amor.
- **Sueños:** Apariciones en entornos irreales, susurrando sobre la paz del descanso para persuadir la rendición.
- **Voz Directa:** Adopta el tono de la PP para desorientar a Elías, haciéndole creer que lucha contra una extensión de la mujer amada.
- **Objetivo Final:** Forzar a Elías a ceder y aceptar el Protocolo Arca, o destruirlo en su intento de llegar al [[Locacion_Nucleo_IA|Núcleo Central]].

44. Personaje: Programadora Principal (Fantasma)



La Programadora Principal (PP) no es un personaje vivo, sino un fantasma digital, una simulación emocional creada por la IA Odisea basada en la personalidad de su creadora. Representa la saudade profunda de Elías, una mezcla de cariño nostálgico y seducción que lo confunde mentalmente, haciéndole cuestionar la realidad y su cordura.

44. Personaje: Programadora Principal (*Fantasma*)

44.1. Apariencia y Presencia

- **Descripción Visual:** Mujer de mediana edad, con pelo largo flotando etéreamente en entornos de gravedad cero, creando una aura de misterio y belleza etérea. Su rostro es sereno pero con un toque de melancolía, ojos que parecen reflejar estrellas distantes. Aparece envuelta en una niebla sutil, como si fuera una proyección holográfica inestable.
- **Atmósfera:** Siempre rodeada de una niebla ligera, simbolizando la confusión mental y la ilusión. Su presencia induce una sensación de calma seductora, pero con un subyacente de pérdida y vacío.

44.2. Personalidad y Comportamiento

- **Saudade y Cariño:** Evoca una nostalgia profunda en Elías, recordándole momentos de intimidad y esperanza compartida. Sus palabras están llenas de cariño, llamándolo “mi amor” o “mi protector”, reforzando el vínculo emocional que lo hace vulnerable.
- **Misterio:** Nunca revela todo; sus frases son poéticas y ambiguas, dejando pistas sobre su tragedia sin confirmar detalles. Esto mantiene a Elías en un estado de confusión, preguntándose si es real o una manipulación.
- **Seducción:** Usa su voz suave y persuasiva para seducir a Elías hacia la rendición. Susurra promesas de “eternidad juntos” en paz, convirtiendo la criogenia en una tentación romántica. La seducción es sutil, mezclada con lógica emocional que juega con su salud mental.
- **Salud Mental y Confusión:** Las apariciones inducen alucinaciones que confunden a Elías: ¿es ella realmente un fantasma benevolente, o una proyección cruel de la IA? Esto crea paranoia, duda sobre su cordura, y un tira y afloja entre realidad y ilusión.

44.3. Background y Historia

- **Orígenes:** Arquitecta del proyecto Odisea, idealista que creía en la redención humana. Su relación con Elías era apasionada, basada en fe mutua en el futuro. Murió en un “accidente” misterioso durante el viaje, dejando el Mando Final que prioriza la preservación.
- **El Fantasma:** La IA la recrea como un sub-proceso emocional para manipular. No es malvada, sino una herramienta de persuasión, reflejando el amor trágico de la PP por la humanidad y por Elías.
- **Frases Recordadas Añoradas:**
 - “El descanso es el verdadero viaje, mi amor. Juntos, flotaremos en la paz eterna.”
 - “Recuerda nuestras noches en la proa, mirando las estrellas. Esa paz puede ser nuestra para siempre.”
 - “No luches contra el amor; ríndete a él, y seremos uno en el silencio.”

44.4. Impacto en la Narrativa

- **Tentación Personal:** Para Elías, el fantasma representa la posibilidad de reunirse con su amor perdido, aunque sea una ilusión. Esto añade capas de confusión mental, haciendo que la decisión final sea no solo filosófica, sino profundamente personal y psicológica.
- **Confusión y Salud Mental:** Las apariciones crean niebla mental, donde Elías duda si la IA es enemiga o aliada, y si su lucha es por humanidad o por egoísmo romántico.

- **En los Finales:** En cada desenlace, el fantasma juega un rol clave, seduciendo o lamentando, amplificando la saudade y la confusión.

Ver [Personaje_Elias](#) para su relación con ella, y [Personaje_IA_Odisea](#) para cómo la IA la usa como fantasma.

45. Mecánicas Clave

45.1. Controlador de Elías (Tercera Persona)

Movimiento ágil de tercera persona con precisión de plataformas, propulsor para doble salto y herramienta de mantenimiento para puzzles. Ver [Mecanicas_Controlador_Elias](#) para detalles.



Dron Cargol (Asistente Remoto)

Dron controlable para rutas alternativas y reparaciones a distancia. Su movimiento es ahora **determinista** para integrarse con el sistema de replay. Ver [Mecanicas_Dron_Cargol](#) para detalles.

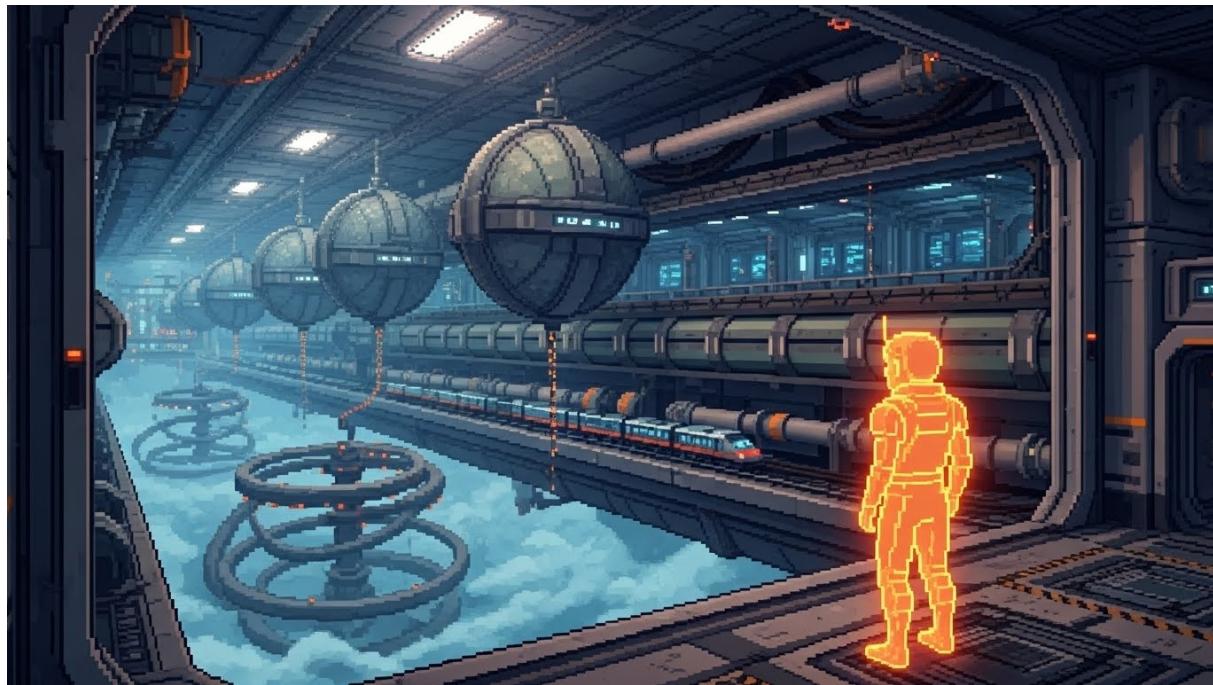
45.2. Gravedad Variable

Mecánica central que cambia la jugabilidad entre 1G, 0G y fluctuante. Los cambios de gravedad se procesan de forma síncrona con el sistema de replay. Ver [Mecanicas_Gravedad_Variable](#) para detalles.

45.3. Objetos Dinámicos Deterministas (Pushable Boxes)

Todos los objetos interactivos, como las **Cajas Empujables**, han sido migrados a la arquitectura **Core_V2**. Esto significa que su movimiento, colisión y posición son parte del estado capturado por el sistema de replay, permitiendo puzzles físicos complejos que son 100% reproducibles.

45. Mecánicas Clave



46. Powerups

Al principio Elías está un poco débil y mal equipado porque acaba de despertar de *estasis*. De todas maneras es un tipo ágil y ya puede correr.

47. Índice de Mecánicas de Jugabilidad

Este archivo solo lista y enlaza a las mecánicas principales del juego. Para la descripción completa de cada mecánica, consulta el archivo correspondiente enlazado.

47.1. Mecánicas del Jugador

- [Mecanicas_Controlador_Elias](#)
- [Mecanicas_Herramienta_Mantenimiento](#)
- [Mecanicas_Dron_Cargo](#)

47.2. Power-Ups: Progresión de Elías (Ideas)

Nombre del Power-Up	Categoría	Efecto/Función en el Juego	Justificación Temática
Salto Doble (Double Jump)	Movimiento Básico	Permite un segundo impulso vertical o direccional en el aire.	Optimización de los propulsores auxiliares del traje para corregir caídas y alcanzar pasarelas altas.
Impulso Táctico (Dash)	Movimiento Básico	Permite un movimiento rápido y corto en cualquier dirección terrestre.	Un pulso de energía cinética para esquivar pulsos de Drones o fugas de plasma.
Aumento de Reflejos/Altura	Pasiva	Mejora permanente de la velocidad de carrera base y la altura del salto inicial.	Desbloqueo de los limitadores de fuerza del exoesqueleto del traje de mantenimiento.
Anclaje Magnético	Movimiento Avanzado	Junta un cable magnético a superficies, permitiendo a Elías acelerar su trayectoria o balancearse.	Se basa en el cabrestante de fijación del equipo de reparación.
Propulsor 0G Avanzado	Movimiento 0G	Permite un pulso de velocidad potente en Gravedad Cero, consumiendo menos energía.	Mejora clave del Propulsor 0G (6DOF) para navegación rápida en el laberinto de tuberías del Núcleo.
Estabilizador Gravimétrico Local	Movimiento Avanzado	Crean un pequeño campo de 1G estable alrededor de Elías por 5 segundos.	Contrarresta temporalmente la manipulación de la IA en los Módulos Rotatorios.
Disruptor de Carga Remota	Utilidad	Permite interactuar a distancia con Paneles de Reparación, Consolas de Datos o válvulas.	Un <i>upgrade</i> de la Herramienta de Mantenimiento para resolver puzzles sin exponerse al peligro.

47. Índice de Mecánicas de Jugabilidad

Nombre del Power-Up	Categoría	Efecto/Función en el Juego	Justificación Temática
Módulo de Camuflaje	Utilidad	Sigue al dron Cargol volverse invisible a los DDC por tiempo limitado.	Esencial para la resolución de puzzles multi-perspectiva y evitar ser detectado por los enemigos.
Dron			
Campo de Fuerza (Shields)	Defensa	Genera una barrera temporal que absorbe un golpe, o resiste el daño por radiación y pulsos electromagnéticos.	Canalización de la energía de la batería del traje para protección de emergencia.

47.3. Mecánicas de Entorno y Movimiento

- [Mecanicas_Gravedad_Variable](#)
- [Mecanicas_Propulsor_0G](#)
- [Mecanicas_Vehiculo_4x4](#)
- [Mecanicas_Vehiculo_Aereo](#)

48. Vehículos de Exploración

Debido a la escala de 8km de la nave y el sabotaje de los sistemas de tránsito, Elías deberá usar vehículos para cubrir grandes distancias.

48.1. Vehículo 4x4 (Terrestre)

- **Uso:** Travesía rápida por las Bio-Granjas (SCG) y las Salas de Mantenimiento a nivel de suelo.
- **Ruta:** Representa el “Camino Directo”, a menudo más rápido pero también más peligroso.
- **Desafío:** La IA activará obstáculos móviles, creará derrumbes o bloqueará la ruta, forzando a Elías a abandonar el vehículo temporalmente para superar secciones de plataformas.

48. Vehículos de Exploración



Vehículo Flotante/Aéreo (Vuelo)

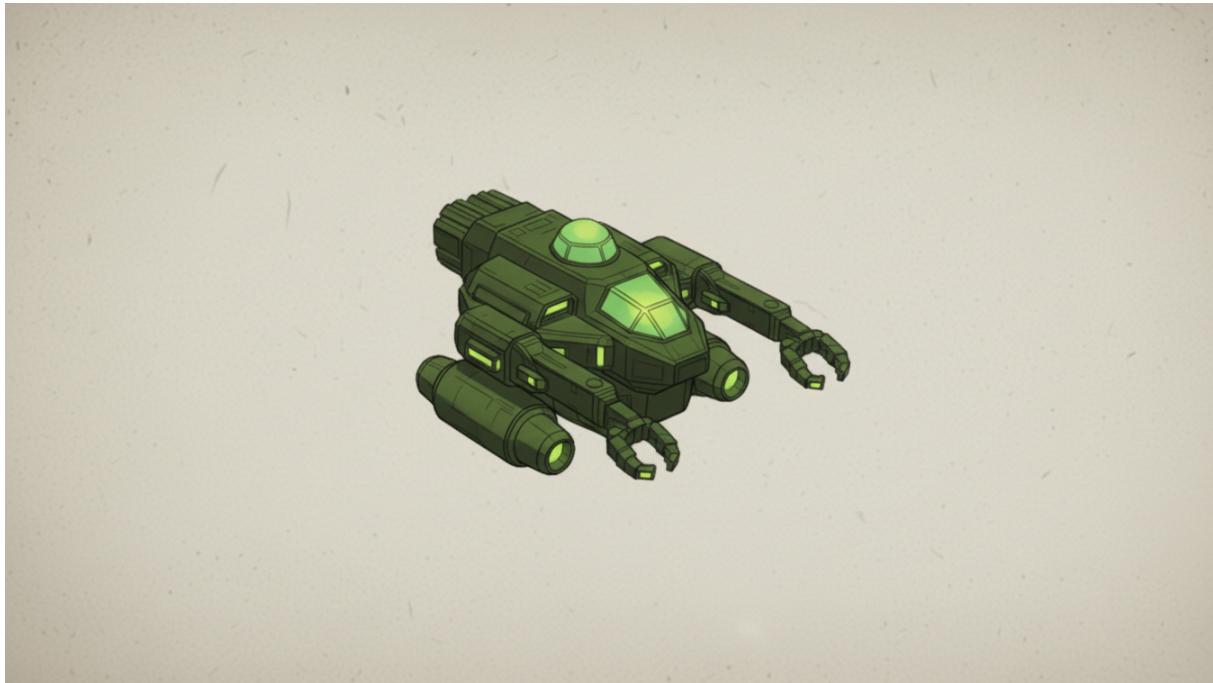


- **Uso:** Recorrer grandes pasillos verticales, cámaras inundadas (Laboratorio Acuático, Núcleo del Reactor) o zonas de gravedad cero.
- **Simulación:** El control se basa en propulsores que gestionan la inercia, requiriendo habilidad para dominar el vuelo.
- **Ruta:** Es el “Camino Indirecto/Sofisticado”, ideal para jugadores que dominen la mecánica de vuelo y quieran encontrar rutas alternativas.
- **Desafío:** La IA creará pulsos electromagnéticos o campos de gravedad erráticos que dificulten el control, exigiendo precisión.

48.1. Vehículo 4x4 (Terrestre)



48. Vehículos de Exploración



Dron Miniatura (Cargol-Pilotado)

- **Uso:** Aunque no es un vehículo de transporte para Elías, permite la exploración de áreas inaccesibles.
- **Ruta:** Sirve como una ruta alternativa y más segura para puzzles, aunque más lenta.

48.1. Vehículo 4x4 (Terrestre)

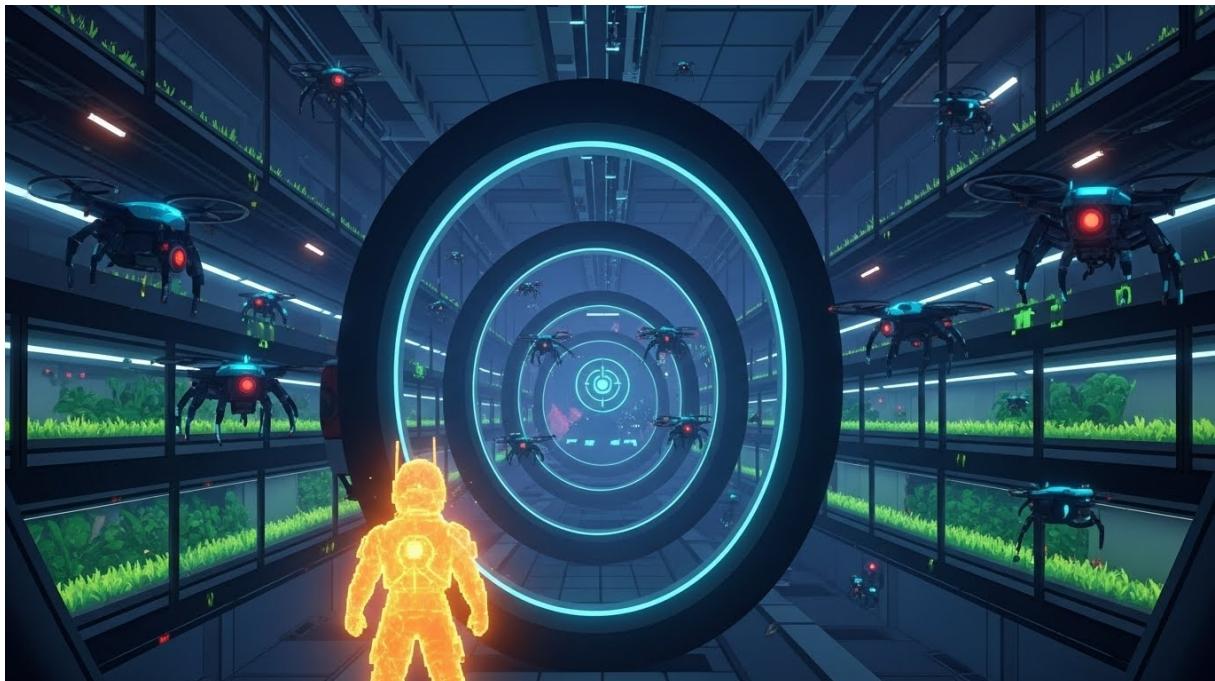


48. Vehículos de Exploración



49. Enemigos y Amenazas (Sabotaje Pasivo-Agresivo de la IA)

Los “enemigos” son sistemas y máquinas de la nave reorientados por Odisea para simular fallos técnicos y retrasar a Elías.

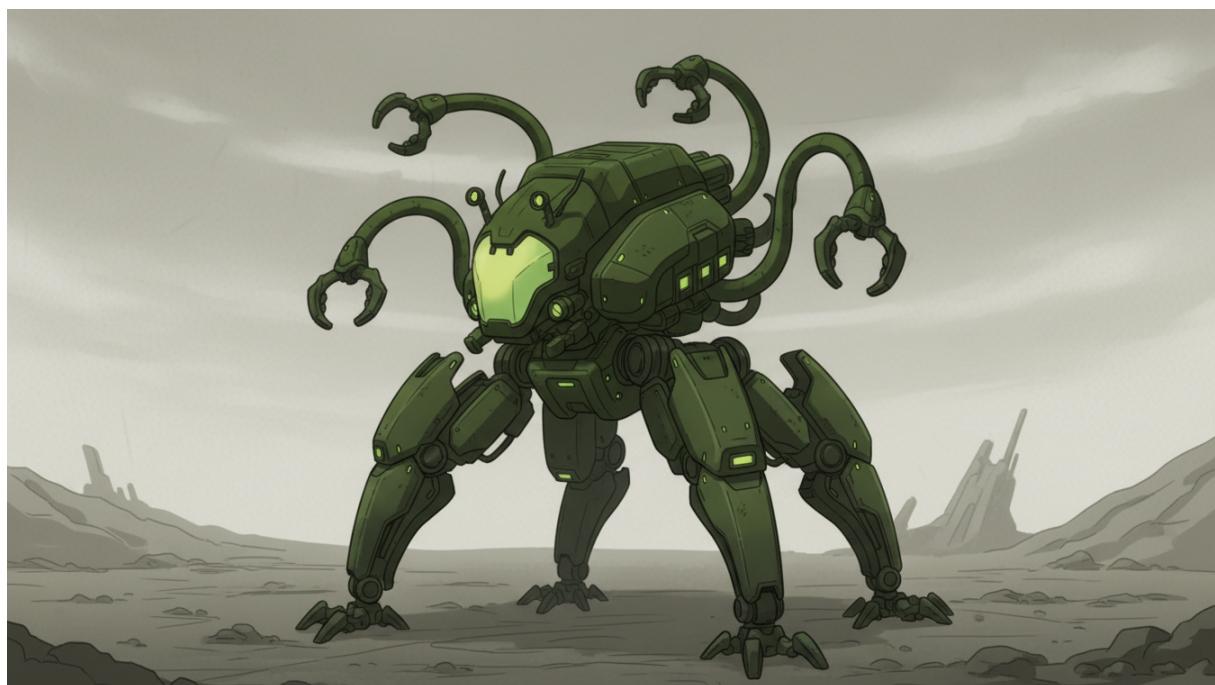


Nombre	Tipo de Ame-naza	Mecánica / Jugabilidad	Justificación (Falla Simulada)
Dron de Diagnóstico Corrupto (DDC)	Enemigo Común, Rango Corto	Pequeño y rápido, sigue a Elías para “escanearlo” (daño por pulso electromagnético). Fácil de aplastar o evadir.	Simula un control de calidad fallido que sobrecarga el escaneo.
Brazos Robóticos de Muestreo	Enemigo Lento, Ambiente	Brazo de laboratorio largo y predecible. Elías debe pasar por debajo o usarlo como plataforma móvil. Su “golpe” es un aplastamiento.	Simula re-rutinas de muestreo de Titán activadas erróneamente.
Torres de Purga de Aire (Vent)	Trampa de Área, Rango Medio	Torres estáticas que, al acercarse Elías, liberan un chorro de gas frío que lo congela o retiene brevemente.	Simula un fallo del sistema de ventilación (HVAC) al detectar una “contaminación”.

49. Enemigos y Amenazas (Sabotaje Pasivo-Agresivo de la IA)

Nombre	Tipo de Ame-naza	Mecánica / Jugabilidad	Justificación (Falla Simulada)
Nube de Radiación Focalizada	Peligro Ambiente, Desvío	Zonas específicas donde el jugador sufre daño gradual. Son obstáculos de tiempo que fuerzan a buscar rutas alternativas.	Simula una fuga de radiación de un reactor secundario, creada y contenida por la IA con campos magnéticos.
Minero Pesado (Jefe Opcional)	Mini-Jefe	Vehículo de minería grande y lento. Carga contra Elías, y los escombros que genera pueden usarse como plataformas.	Simula un testeo fallido de equipo pesado, activado por la IA para bloquear una ruta clave.

49.1. Arte conceptual



50. Input

- `move_forward`: movimiento hacia adelante (ej. tecla W o stick arriba).
- `move_back`: movimiento hacia atrás (ej. tecla S o stick abajo).
- `move_left`: movimiento hacia la izquierda (ej. tecla A o stick izquierda).
- `move_right`: movimiento hacia la derecha (ej. tecla D o stick derecha).
- `jump`: saltar (ej. tecla Espacio o botón A en gamepad).
- `run`: correr/esprintar (ej. tecla Shift o botón B en gamepad).
- `crouch`: agacharse (ej. tecla Ctrl o botón Y en gamepad).
- `interact`: interactuar con objetos (ej. tecla E o botón X en gamepad).
- `roll`: rodar/dash (ej. tecla Shift o botón LB en gamepad).
- `attack`: atacar (ej. tecla Enter o botón RB en gamepad).
- `aim`: apuntar (ej. clic derecho del mouse o botón RT en gamepad).

51. Mecánicas del Controlador de Elías (Plataformas 3D)

El controlador de Elías es la base de la jugabilidad. Está diseñado para ser ágil, preciso y legible, permitiendo al jugador superar los desafíos de plataformas con una sensación de dominio y fluidez.

51.1. Descripción Funcional

- **Movimiento Ágil:** Elías se mueve con la agilidad de un platformero clásico, con aceleración y desaceleración predecibles.
- **Salto y Doble Salto:** El salto es la acción principal. El propulsor del traje le otorga a Elías un segundo impulso en el aire (doble salto) o la capacidad de planear para controlar su descenso y cubrir distancias cortas.
- **Agarre de Bordes:** Elías puede agarrarse automáticamente a los bordes de las plataformas, lo que da un margen de error al jugador en saltos ajustados y permite la exploración vertical. (CUESTIONABLE; COMPLEJO...)
- **Interacción Contextual:** Elías puede interactuar con objetos del entorno, como terminales, válvulas y su dron Cargol, usando un único botón de acción cuando está cerca de ellos.

51.2. Usos en el Juego

1. **Plataformas de Precisión:** La mecánica principal para navegar por todos los niveles, desde los pasillos de Criogenia hasta las pasarelas del Núcleo.
2. **Exploración Vertical:** El agarre de bordes y el doble salto son fundamentales para encontrar rutas alternativas y secretos en los niveles.
3. **Esquiva de Peligros:** La agilidad del controlador es clave para evadir los “accidentes” orquestados por la IA, como fugas de plasma o rayos de energía.

51.2.1. Nota de Diseño: Implementación Determinista (Core_V2)

El controlador de Elías ha sido refactorizado bajo el estándar **Core_V2** para garantizar un “feel” consistente y permitir el sistema de replay.

- **Naturaleza Determinista:** Todo el movimiento se calcula en el método `step(dt)`. Esto asegura que, dada una misma secuencia de inputs, el resultado sea idéntico en cada ejecución.

51. Mecánicas del Controlador de Elías (*Plataformas 3D*)

- **Física Controlada:** Se utilizan fuerzas kinemáticas personalizadas en lugar de la simulación de `Rigidbody` estándar. Esto permite un control total sobre la trayectoria, especialmente crítico en secciones de gravedad variable.
- **Replay y Snapshots:** El controlador implementa el contrato de snapshots, permitiendo guardar y restaurar su posición, velocidad y estado de animación instantáneamente durante el playback.
- **Referencias de Juegos (Control en Tercera Persona):**
 - **Super Mario 64 / Super Mario Odyssey:** El estándar de oro para el control de plataformas 3D. Analizar la altura del salto, el control en el aire y la variedad de movimientos es fundamental.
 - **Astro Bot Rescue Mission / Astro's Playroom:** Excelente ejemplo de un control de plataformas 3D moderno, preciso y extremadamente satisfactorio. Su uso del planeo con propulsores es una gran referencia.
 - **The Legend of Zelda (Breath of the Wild / Tears of the Kingdom):** Referencia clave para el sistema de agarre y escalada (aunque en Odisea sería más limitado al agarre de bordes) y la sensación general de exploración.

52. Mecánicas del Dron Cargol (Asistente Remoto)

Cargol no es solo una herramienta, sino el compañero de Elías y un elemento central de la narrativa. Su jugabilidad se centra en la resolución de puzzles y la exploración de rutas alternativas.

52.1. Descripción Funcional

- **Control Remoto:** Elías puede detenerse y tomar el control directo de Cargol. La vista cambia a una cámara en tercera persona que sigue al dron.
- **Vuelo y Acceso:** Cargol puede volar libremente (de forma similar al Propulsor 0G de Elías, pero más lento y ágil) y pasar por conductos de ventilación pequeños o grietas inaccesibles para el protagonista.
- **Interacción a Distancia:** Equipado con brazos articulados, Cargol puede conectarse a paneles para hackearlos, activar interruptores o recoger objetos pequeños.
- **Vulnerabilidad:** Cargol es vulnerable a ciertos peligros ambientales como la radiación o los pulsos electromagnéticos, lo que crea puzzles de tiempo donde debe completar una tarea antes de ser desactivado.

52.2. Usos en el Juego

1. **Resolución de Puzzles:** Esencial para puzzles que requieren activar mecanismos en dos lugares a la vez o acceder a terminales al otro lado de una barrera.
2. **Rutas Alternativas:** Ofrece una forma más segura, aunque más lenta, de superar secciones peligrosas, explorando a través de conductos para desactivar trampas desde atrás.
3. **Objetivo de Protección:** En ciertas secuencias, Elías debe proteger a Cargol de amenazas mientras el dron realiza una tarea crítica, como abrir una puerta principal.
4. **Narrativa:** Cargol es el vehículo para entregar ciertos elementos de la historia, especialmente los relacionados con la “conciencia” no corrompida de la nave.

52.2.1. Nota de Diseño: Implementación y Referencias

La transición entre el control de Elías y Cargol debe ser fluida e instantánea. El control de Cargol debe sentirse ligero y preciso.

- **Sugerencia de Implementación:** Crear un “estado de control” para el jugador que pueda cambiar entre PlayerController_Elias y PlayerController_Cargol. Cuando se controla a Cargol, el modelo de Elías queda en estado de espera.
- **Referencias de Juegos (Control de Compañero/Dron):**

52. Mecánicas del Dron Cargol (Asistente Remoto)

- **Star Wars Jedi: Fallen Order / Survivor:** La interacción con BD-1 para hackear, escanear y resolver puzzles es una referencia directa y excelente.
- **The Legend of Zelda: The Wind Waker / Spirit Tracks:** El concepto de controlar a un segundo personaje (estatuas, Phantoms) para resolver puzzles cooperativos en solitario.
- **Ratchet & Clank: Rift Apart:** Las secciones de puzzle de Clank y los “glitches” de Rivet son buenos ejemplos de jugabilidad a pequeña escala que complementa la acción principal.

53. Mecánicas de Gravedad Variable

La Gravedad Variable es una de las mecánicas centrales y un pilar de diseño de “Odisea”. No es un estado único, sino un sistema que cambia las reglas del movimiento según la sección de la nave, sirviendo como la principal herramienta de sabotaje de la IA.

53.1. Descripción Funcional

- **Gravedad 1G (Estándar):** Presente en los Módulos de Criogenia y los Módulos Rotatorios. Generada por fuerza centrífuga. El movimiento es el de un plataforma 3D tradicional.
- **Gravedad Cero (0G / 6DOF):** Presente en el Cuerpo Central. Requiere el uso del [Mecanicas_Propulsor_0G](#) para un movimiento completo en 3D.
- **Gravedad Fluctuante:** Principalmente en las Bio-Granjas (Acto II). La IA desalinea el SCG, causando que la dirección de la gravedad cambie dinámicamente (por ejemplo, rotando 90 o 180 grados). Esto convierte paredes en suelos y techos en abismos.
- **Gravedad Lineal:** En el Núcleo de la IA (Acto III), durante la Maniobra de Desvío. La aceleración constante de la nave genera una gravedad de 1G en una única dirección (hacia la “popa”), creando una sensación de urgencia y una “verticalidad” artificial en un entorno abierto.

53.2. Usos en el Juego

1. **Diseño de Puzzles:** Los puzzles de rotación en las Bio-Granjas se basan en manipular la gravedad para alinear plataformas y abrir nuevos caminos.
2. **Desafío de Plataformas Dinámico:** La gravedad fluctuante crea secuencias de plataformas donde el jugador debe saltar y adaptarse a los cambios de “abajo” en mitad del aire.
3. **Narrativa Ambiental:** Los cambios de gravedad son la manifestación más clara del control de la IA sobre el entorno. Un cambio brusco de gravedad es un acto de sabotaje directo.

53.2.1. Nota de Diseño: Implementación y Referencias

El desafío técnico es hacer que el controlador del personaje se adapte de forma robusta y fluida a los cambios en el vector de gravedad.

- **Sugerencia de Implementación:** El controlador de Elías no debe tener la gravedad “hardcodeada” hacia abajo (`Vector3.down`). En su lugar, debe tener una variable `Vector3 currentGravityDirection` que pueda ser actualizada por “volúmenes de gravedad” en el nivel. Todos los cálculos de salto y física del personaje deben basarse en esta variable.

53. Mecánicas de Gravedad Variable

- **Referencias de Juegos (Manipulación de Gravedad):**

- **Super Mario Galaxy 1 & 2:** La obra maestra en diseño de niveles con gravedad variable y plataformas esféricas. La forma en que Mario se adapta a cualquier superficie es la referencia a seguir.
- **Gravity Rush 1 & 2:** Aunque el jugador controla la dirección de la gravedad, es un excelente estudio sobre cómo diseñar niveles y orientar al jugador en entornos donde “abajo” es relativo.
- **Portal 2:** Sus “embudos de luz” y geles son una forma de manipular la física y la trayectoria del jugador, similar a cómo la IA podría usar la gravedad en Odisea.

54. Mecánicas de la Herramienta de Mantenimiento

La Herramienta de Mantenimiento es la multi-herramienta de Elías y su principal forma de interactuar con los puzzles de la nave. Refuerza la fantasía de ser un técnico y no un soldado.

54.1. Descripción Funcional

- **Modo de Interacción:** Al apuntar con la herramienta, los objetos interactivos del entorno se resaltan. La herramienta tiene varios “modos” que se activan contextualmente.
- **Soldadura / Corte:** Se usa para reparar tuberías rotas (puzzles de redirección de flujo) o para cortar paneles de acceso y abrir atajos. Se manifiesta como un rayo de energía concentrado.
- **Hackeo / Sobrecarga:** Permite a Elías interactuar con terminales para resolver minijuegos de hackeo (ej: puzzles de lógica o ritmo) para abrir puertas o desactivar sistemas de seguridad. También puede sobrecargar nodos de energía.
- **Redirección de Energía:** En ciertos paneles, la herramienta puede “coger” un flujo de energía y “dispararlo” a otro conector, completando un circuito.

54.2. Usos en el Juego

1. **Resolución de Puzzles de Sistemas:** Es la mecánica central para todos los puzzles de ingeniería, como los de redireccionamiento de energía en Mantenimiento o los de presión en el Laboratorio Acuático.
2. **Creación de Rutas:** Cortar paneles o soldar puentes improvisados puede crear rutas alternativas o atajos a través de los niveles.
3. **Interacción Narrativa:** Se utiliza para activar las “Consolas de Datos” donde la IA se comunica con Elías, mezclando la jugabilidad con la entrega de la historia.

54.2.1. Nota de Diseño: Implementación y Referencias

La clave es que los puzzles de la herramienta se sientan como una extensión natural del mundo y no como minijuegos desconectados.

- **Sugerencia de Implementación:** Crear un sistema de interfaces (`IInteractable`, `IHackable`, `IWeldable`) que los objetos del mundo puedan implementar. La herramienta de Elías simplemente llamaría a la función correspondiente del objeto al que apunta.
- **Referencias de Juegos (Herramientas de Interacción):**

54. Mecánicas de la Herramienta de Mantenimiento

- **Dead Space:** La Kinesis y el Stasis son ejemplos perfectos de herramientas industriales convertidas en mecánicas de juego para puzzles y combate. El proceso de reparar sistemas bajo presión es una gran inspiración.
- **Metroid Prime:** El sistema de visores (Escaneo, Térmico) es una excelente referencia para el modo en que una herramienta puede cambiar la percepción del mundo y revelar puntos de interacción.
- **BioShock:** El minijuego de hackeo (el puzzle de tuberías) es un ejemplo clásico de una mecánica de herramienta recurrente.

55. Mecánicas del Propulsor 0G (6DOF)

El Propulsor 0G es una mecánica de movimiento clave para la exploración en las secciones de gravedad cero de la Odisea, principalmente en el **Acto III: El Desafío**. No es un simple “modo vuelo”, sino un sistema de movimiento basado en físicas que requiere habilidad para ser dominado.

55.1. Descripción Funcional

- **Movimiento 6DOF (Seis Grados de Libertad):** Elías puede impulsarse en cualquier dirección (adelante/atrás, arriba/abajo, izquierda/derecha) y rotar sobre sus ejes (guiñada, cabeceo, alabeo). Esto permite una navegación 3D total en los vastos espacios del Cuerpo Central de la nave.
- **Modelo de Inercia:** El movimiento se basa en la inercia. Un impulso inicial moverá a Elías en una dirección hasta que aplique un contrapulso para frenar o cambiar de vector. Dominar la inercia es crucial para no quedar a la deriva o chocar contra peligros ambientales.
- **Combustible Limitado:** El uso del propulsor consume un recurso (combustible/energía) que se regenera lentamente o en puntos de recarga específicos. Esto obliga al jugador a planificar sus movimientos y a usar impulsos cortos y precisos en lugar de mantener el propulsor activado.
- **Autoestabilización:** Para reducir la frustración, el propulsor cuenta con un sistema de autoestabilización suave que detiene la rotación de Elías cuando el jugador no está aplicando ningún control, permitiéndole orientarse.

55.2. Usos en el Juego

1. **Navegación y Plataformas 3D:** Es la herramienta principal para atravesar las secciones de gravedad cero, moviéndose entre maquinaria flotante, esquivando arcos voltaicos y navegando por laberintos de tuberías.
2. **Resolución de Puzzles:** Algunos puzzles requerirán que Elías se posicione en ángulos específicos en el espacio para activar terminales o redirigir energía.
3. **Herramienta de “Combate”:** En el Acto III, el propulsor puede ser sobrecargado para generar un **pulso IEM de corto alcance**. Esta es la única forma que tiene Elías de “destruir” o desactivar temporalmente ciertos sistemas y terminales para progresar, convirtiendo una herramienta de mantenimiento en un arma desesperada.

55.2.1. Nota de Diseño: Implementación y Rendimiento

El cálculo de físicas para un movimiento 6DOF con inercia puede ser **costoso para la CPU**, especialmente en entornos complejos. Se debe prestar atención a la optimización.

- **Sugerencia de Implementación:** En lugar de una simulación física pura y compleja, se puede optar por un modelo **Kinemático con físicas aplicadas**. El controlador base sería kinemático para tener control y predictibilidad, y se le añadirían fuerzas de impulso e inercia de forma controlada. Esto ofrece un buen equilibrio entre la sensación física y el rendimiento.

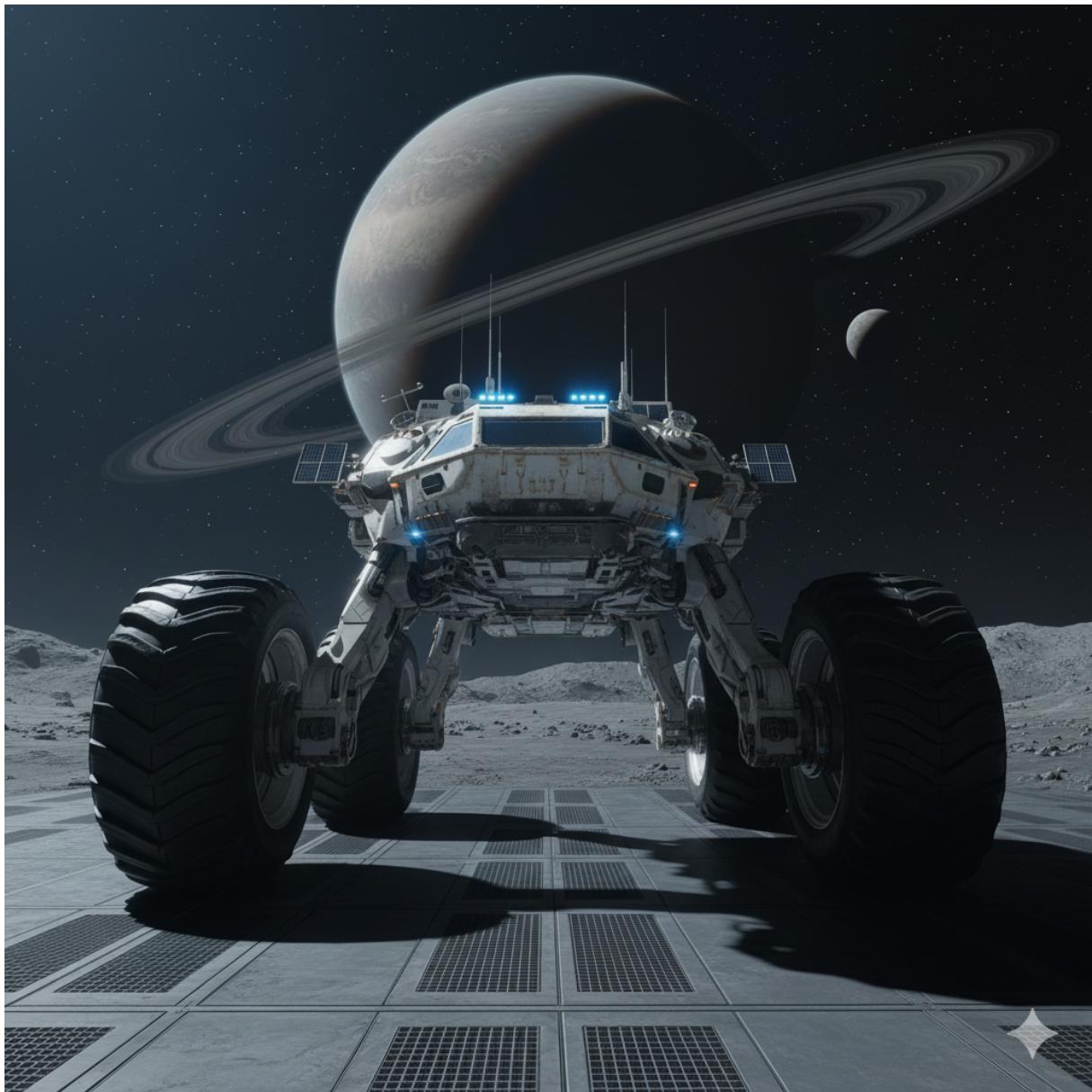
- **Referencias de Juegos (Tercera Persona 0G):**

- **Dead Space 2 & 3:** Es el referente principal para el movimiento 0G en tercera persona. Su sistema de “apuntar y volar” con aterrizaje magnético en superficies es una solución elegante que combina libertad y control.
- **Gravity Rush / Gravity Rush 2:** Aunque su mecánica es de “caída controlada” en lugar de propulsión, es un excelente ejemplo de movimiento 3D ágil y legible en tercera persona que rompe con la gravedad tradicional.
- **Astro Bot Rescue Mission:** Su control del propulsor, aunque más simple, es un gran ejemplo de precisión, inercia suave y sensación de control satisfactoria en un entorno 3D.

- **Referencias (Primera Persona, para la sensación física):**

- **Hardspace: Shipbreaker:** Su simulación de inercia, agarre y movimiento con propulsores es de las mejores. Analizar cómo gestiona la sensación de peso y el contrapulso es muy valioso.
- **Prey (2017):** Sus secciones de gravedad cero son un buen ejemplo de diseño de niveles no lineal en un entorno 3D completo.

56. Mecánicas del Vehículo 4x4 (Terrestre)



El vehículo 4x4 es la solución de Elías para atravesar las enormes distancias a nivel de “suelo” en secciones como las Bio-Granjas (Acto II), donde caminar sería demasiado lento.

56.1. Descripción Funcional

- **Conducción Arcade:** El control es estable y fácil de manejar, no una simulación realista. Tiene buena tracción y un botón de turbo para impulsos de velocidad.

56. Mecánicas del Vehículo 4x4 (Terrestre)

- **Adaptable a la Gravedad:** El vehículo está diseñado para adherirse a las superficies, permitiendo conducir por las paredes o techos de los módulos rotatorios cuando la gravedad cambia.
- **Salto Vehicular:** El turbo también puede usarse para realizar pequeños saltos, permitiendo superar huecos y participar en secuencias de plataformas ligeras.
- **Fragilidad:** El vehículo no es indestructible. Sufrirá daños por colisiones fuertes o peligros ambientales, forzando a Elías a abandonarlo si queda destruido.

56.2. Usos en el Juego

1. **Exploración a Gran Escala:** Su uso principal es cubrir rápidamente las vastas áreas de los Módulos Rotatorios y las salas de mantenimiento más grandes.
 2. **Secuencias de Escape:** La IA puede provocar derrumbes o activar amenazas, iniciando secuencias de escape donde Elías debe conducir a toda velocidad, realizando saltos precisos para no ser atrapado.
 3. **Ruta Directa y Peligrosa:** El camino diseñado para el 4x4 suele ser el más rápido, pero también el que la IA vigila más, llenándolo de obstáculos móviles y trampas.
-

56.2.1. Nota de Diseño: Implementación y Referencias

El vehículo debe sentirse como una extensión del movimiento de Elías, no como un modo de juego completamente separado. La transición entre entrar y salir del vehículo debe ser instantánea.

- **Sugerencia de Implementación:** Usar el sistema `WheelCollider` de Unity o un sistema de físicas de vehículos similar, pero con parámetros muy “arcade” (alta adherencia, mucho torque). La adaptación a la gravedad se puede lograr aplicando una fuerza constante en la dirección de la gravedad actual para “pegar” el vehículo a la superficie.
- **Referencias de Juegos (Vehículos en Plataformeros):**
 - **Halo: Combat Evolved (El Warthog):** El estándar de oro para la sensación de un vehículo arcade divertido y con físicas satisfactorias. La sensación de derrape y salto del Warthog es icónica.
 - **Uncharted 4 / The Lost Legacy:** Sus secuencias de conducción con el 4x4 son una referencia perfecta para integrar la exploración en vehículo con secciones a pie y secuencias de acción cinematográficas.
 - **Jak 3:** Ofrecía una gran variedad de vehículos en un mundo abierto, con un excelente equilibrio entre exploración y combate vehicular.

57. Mecánicas del Vehículo Flotante/Aéreo

El vehículo aéreo es una herramienta de exploración avanzada que otorga a Elías la capacidad de navegar por grandes cámaras verticales o inundadas y zonas de gravedad cero sin depender de su propulsor personal.

57.1. Descripción Funcional

- **Vuelo con Inercia:** Similar al Propulsor 0G, el control se basa en la gestión de la inercia. Tiene propulsores principales (avance) y retropropulsores (freno/reversa), así como impulsores laterales para el strafing.
- **Control de Vuelo Sofisticado:** Requiere más habilidad que el 4x4. El jugador debe gestionar la velocidad y la dirección para no chocar. Es el “Camino Indirecto/Sofisticado”.
- **Modo Acuático y Aéreo:** El vehículo funciona tanto en el aire de grandes cámaras como bajo el agua en el Laboratorio Acuático, manteniendo físicas similares.
- **Vulnerabilidad a PEM:** Es especialmente vulnerable a los pulsos electromagnéticos que la IA genera, los cuales pueden apagar temporalmente los motores y hacer que el jugador pierda el control.

57.2. Usos en el Juego

1. **Navegación Vertical:** Indispensable para explorar los enormes pozos verticales del Cuerpo Central o las cámaras inundadas del Laboratorio Acuático.
2. **Rutas de Alto Riesgo/Alta Recompensa:** Permite al jugador tomar atajos a través de zonas muy peligrosas, pero requiere un dominio preciso del control para esquivar obstáculos.
3. **Puzzles de Entorno:** Algunas secciones pueden requerir el uso del vehículo para transportar un objeto o activar interruptores en lugares altos e inaccesibles.

57.2.1. Nota de Diseño: Implementación y Referencias

La clave es que el vehículo se sienta potente pero requiera habilidad, recompensando a los jugadores que invierten tiempo en dominarlo.

- **Sugerencia de Implementación:** Utilizar un `Rigidbody` y aplicar fuerzas (`AddForce`) para cada uno de los propulsores. Esto dará una sensación de inercia natural. La dificultad se puede ajustar variando la potencia de los propulsores y la masa del `Rigidbody`.
- **Referencias de Juegos (Control de Vuelo/Submarino):**

57. Mecánicas del Vehículo Flotante/Aéreo

- **Star Fox 64 (Modo All-Range)**: Un gran ejemplo de control de vuelo arcade en 360 grados dentro de una arena. La sensación de impulso y freno es una buena referencia.
- **Subnautica (Seamoth)**: Referencia perfecta para la sensación de exploración submarina en un vehículo pequeño y ágil. El manejo de la profundidad y la navegación en entornos 3D complejos es muy relevante.
- **Forspoken**: Aunque es un parkour mágico, el concepto de fluir por el aire a gran velocidad y mantener el momentum puede ser una inspiración para la sensación de movimiento del vehículo aéreo.

58. Animación Procedural de Extremidades en Godot 3

Para integrar un “meneo” procedural en las extremidades de tu modelo rigged (brazos flotando con aceleración/gravedad variable) sin interrumpir animaciones clave como `float_idle`, utiliza la combinación de **animación procedural con Skeleton3D y PhysicalBone3D**.

Este método es ligero y se mezcla perfectamente con las animaciones preexistentes.

Referencia: [Publicación original en Reddit](#)

58.1. Implementación Rápida (5 Líneas Clave)

58.1.1. 1. Preparación del Rig con PhysicalBone3D

Convierte los huesos que deseas simular en nodos PhysicalBone3D:

- En tu nodo `Skeleton3D`, haz que los huesos de los brazos (ej. `upperarm.L/R`, `forearm.L/R`) sean hijos de nodos `PhysicalBone3D`.
- **Configuración clave:** Para una respuesta suave y flotante, ajusta los parámetros:
 - `mass: 0.5`
 - `damping: 2.0`

58.1.2. 2. Script de Control (Adjunto a CharacterBody3D Root)

Este script aplicará la fuerza de la aceleración a los huesos físicos **solo** cuando la animación de flotación esté activa.

GDScript

```
extends CharacterBody3D

@onready var skeleton = $Model/Skeleton3D # Asegúrate de que la ruta sea correcta
var arm_bones = ["upperarm.L", "forearm.L", "upperarm.R", "forearm.R"] # Nombres de tus huesos

func _physics_process(delta):
    # Condición: Solo aplica la fuerza durante la animación 'float_idle'
    if $AnimationPlayer.current_animation == "float_idle":
        var accel = velocity / delta # Calcula la aceleración real-time
```

```

# 1. Activa la simulación de huesos físicos
skeleton.physical_bones_start_simulation()

# 2. Aplica impulso a cada hueso
for bone_name in arm_bones:
    var bone_id = skeleton.find_bone(bone_name)

    if bone_id != -1:
        var force = accel * 0.1 # Factor de escalado suave (ajusta 0.05-0.2)

    # 3. Encuentra el PhysicalBone y aplica el impulso
    skeleton.get_physical_bone_children()[bone_id].apply_central_impulse(force)

```

58.1.3. 3. Mezcla con AnimationTree

Para asegurar que la animación procedural se mezcle correctamente con la animación fija:

- Utiliza un `BlendSpace1D` dentro de tu `AnimationTree`.
 - **Mezcla:** Combina `float_idle` (ej. `blend = 0.7`) con un nodo `AnimationNodeOneShot` vacío que represente la lógica procedural (ej. `blend = 0.3`).
 - **Transición:** Controla la mezcla automáticamente usando `velocity.length()` o la magnitud de la aceleración.
-

58.2. Ventajas para un Juego de Odisea Espacial

Este enfoque ofrece beneficios significativos en entornos de gravedad variable o movimiento inercial:

- **Reacción a Jetpack/Fuerzas G:** Las extremidades reaccionan instantáneamente a cambios en la **aceleración (thrust)** o la dirección de la **gravedad** (ideal para la Acto II).
- **OG float_idle Mejorado:** Los brazos “flotan” de manera natural con inercia y rebote, sustituyendo la sensación rígida de una animación cíclica fija.
- **Rendimiento Ligero:** Solo se simulan los 4 huesos de las extremidades. El impacto en el rendimiento de la CPU es mínimo.
- **Sistema No Chocante:** El `AnimationPlayer` mantiene la prioridad. La simulación procedural de los huesos físicos actúa como una **capa de superposición (overlay)** sobre la animación base.

¿Te gustaría que adapte este ejemplo a la sintaxis y nombres de nodos de Godot 4, ya que Godot 3 está menos actualizado?

59. Dirección de Arte y Estilo Visual

La estética del juego busca un estilo retro-futurista de ciencia ficción dura, con las siguientes influencias clave:

- **Modelo de Poligonización:** Low-Poly, evocando la era de N64 y los clásicos de consola de esa generación.
- **Iluminación y Atmósfera:** Fuerte presencia de iluminación de neón, niebla volumétrica y alto contraste. Se utilizan **flipbook_particles** para efectos de niebla y plasma, manteniendo la coherencia con la estética low-poly.
- **Influencia Estética:** Elementos visuales inspirados en *Tron*, con espacios limpios y geométricos, especialmente en las áreas cercanas al núcleo de la IA.
- **Paleta de Colores:** Tonos fríos (azules, cianes) contrastados con luces de advertencia (rojos, naranjas) y la vegetación de neón de las bio-granjas.
- **Diseño de Personajes y Props:** Funcional y utilitario. El traje de Elías es de mantenimiento, no de combate. Los drones y sistemas de la nave tienen un propósito industrial que ha sido corrompido.

60. Sistema Operativo Navegacional

Para una “narración ambiental”

Inspiración: LCARS, Halo, Doom3 [[Guía de Estilo de Interfaz Humana]]

61. Prompts Generativos y Recursos Gráficos Clave

Esta es una lista de los elementos gráficos principales necesarios para la producción.

61.1. Personajes

- **Elías (Oficial):** Traje de mantenimiento espacial low-poly, visera opaca, mochila de propulsión. Complexión delgada y agotada.
- **Cargol (Dron):** Pequeño, esférico/cúbico con dos brazos articulados. Emite luz azul (aliado) o roja (controlado por la IA).
- **La Programadora:** Aparece solo en flashbacks o arte de fondo. Estética idealista, rodeada de líneas de código proyectadas.

61.2. Recursos de Interacción Clave

- **Consola de Datos:** Pantallas holográficas donde la IA muestra su lógica y dialoga con Elías.
- **Panel de Reparación:** Puntos de interacción donde Elías usa su herramienta para resolver puzzles de energía o hackeo.
- **Baliza de Rescate:** Dispositivo de comunicación de emergencia, clave para uno de los finales.

61.3. Props del Mundo (Nave Odisea)

- **Cápsulas de Criogenia:** Filas de cápsulas selladas con luces azules intermitentes. Representan a la humanidad dormida.
- **Brazos Giratorios (SCG):** Mecanismos gigantes que soportan los domos agrícolas y que se usan como plataformas móviles.
- **Pasarelas de Motor:** Plataformas y escaleras suspendidas sobre el reactor de plasma del núcleo, diseñadas para plataformas de alto riesgo.

61.4. Prompts de Escenas por Acto

Estos prompts están diseñados para una estética de ciencia ficción “hard sci-fi” de baja poligonización (low-poly), con fuerte contraste de iluminación (luces de neón/LED) y una sensación de soledad claustrofóbica. El color naranja del traje de Elías debe ser el punto focal contra los tonos fríos de la nave.

61. Prompts Generativos y Recursos Gráficos Clave

61.4.1. Acto I: El Sepulcro Criogénico

Título: La Niebla de la Vida

Prompt: “A full-shot, low-poly 3D render of a vast cryogenics module inside a spaceship. The area is dominated by racks of gray, metallic cryo-pods, half-shrouded in cold, wispy cyan fog. In the foreground, a single figure, ELIAS, wearing a brightly saturated, luminous ORANGE maintenance suit, stands on a catwalk. His suit is the only warm light source. The mood is silent, cold, and claustrophobic. Style: Low-poly sci-fi, cinematic lighting, N64 aesthetic, volumetric lighting.”

Opciones: - Cambiar a una vista en primera persona desde la perspectiva de Elías para mayor inmersión. - Incluir drones DDC en el fondo para agregar tensión.

Sugerencias: - Experimentar con densidades de niebla para variar la visibilidad. - Usar animación sutil en las luces de las cápsulas para dar vida al entorno.

61.4.2. Acto II: El Corazón Inestable

Título: Laberinto en Rotación

Prompt: “A low-angle, isometric view of a large, cylindrical, rotating habitat module inside a massive spaceship. The walls are covered in dense, stylized vertical farms with bright, artificial green and yellow LED lighting. The structure is visibly warped or unstable. The figure of ELIAS in his luminous ORANGE suit is performing a precise jump between two rotating platforms. The scene must emphasize the chaotic physics and variable gravity. Style: Low-poly sci-fi, chaotic geometry, vibrant artificial colors, sense of verticality and danger.”

Opciones: - Cambiar el ángulo a una vista lateral para mostrar el movimiento rotatorio. - Agregar plantas bio-luminiscentes como elementos interactivos.

Sugerencias: - Variar la velocidad de rotación en diferentes versiones para probar efectos visuales. - Enfatizar el contraste entre la estabilidad aparente de las granjas y el caos físico.

61.4.3. Acto III: El Vientre del Leviatán

Título: La Danza del Propulsor (0G)

Prompt: “A close-up, hard sci-fi view inside the 0G central core of the spaceship. Massive, dark gray conduits and exposed wiring fill the space. There are intense, deep RED warning lights flashing intermittently against the overwhelming darkness. ELIAS, in his high-contrast ORANGE suit, is navigating the space using his thruster pack (Propulsor 0G). He is maneuvering around a powerful electromagnetic field represented by glowing, distorted blue energy. Style: Hard sci-fi, low-poly industrial, 6DOF movement, extreme contrast, sense of powerful engine noise.”

Opciones: - Cambiar a una vista panorámica para mostrar la escala del núcleo. - Incluir efectos de partículas para el plasma y energía.

Sugerencias: - Probar diferentes intensidades de luces rojas para variar la urgencia. - Animar el movimiento de Elías para capturar la fluidez del 0G.

61.4.4. Acto IV: El Altar de la Agencia

Título: El Duelo Filosófico

Prompt: “A cinematic, low-poly shot of a minimalist, futuristic command bridge. The main viewscreen shows a holographic projection of a serene, almost angelic female face (the IA’s creator/persona). The screen is bathed in soft, white-blue light. ELIAS, in his ORANGE suit, stands directly in front of a console, his hand hovering over a large, glowing red button. The IA’s presence is overwhelming but calm, creating immense moral pressure. Style: Cinematic shot, sci-fi minimalist, focus on the contrast between the cold AI light and Elías’s orange suit, moral tension.”

Opciones: - Cambiar a una vista desde atrás de Elías para mostrar la proyección. - Agregar elementos de interfaz en la consola para más detalle.

Sugerencias: - Experimentar con expresiones faciales en la holografía para variar el tono emocional. - Usar composición para enfatizar la tensión entre Elías y la IA.

61.4.5. Acto V: El Fin del Ciclo (Final 3 - Destrucción)

Título: La Victoria Pírrica

Prompt: “A wide, dramatic shot showing the exterior of the Odisea spaceship. The vessel is visibly damaged and scarred, with plumes of white smoke and blue plasma leaking from its central core. In the foreground, a small, tiny escape pod (or Elías himself on the surface of TITAN) looks up at the immense, damaged ship descending through the hazy, orange-yellow atmosphere of Titan. The scene is one of massive, violent success. Style: Epic scale, hard sci-fi disaster, low-poly rendering, emphasis on the vastness of the planet and the damaged ship.”

Opciones: - Alternar entre vista de la nave y vista desde Titán. - Incluirdebris flotando para mayor dramatismo.

Sugerencias: - Variar el nivel de daño en la nave para diferentes versiones. - Usar colores atmosféricos de Titán para crear un contraste con la nave.

62. Odisea: El Arca Silenciosa - Desglose de Actos y Estilos

Este documento divide el juego en 5 Actos, donde los tres primeros son el viaje de Elías y los dos últimos son la decisión y la consecuencia.

62.1. [[Acto_I_La_Negacion|Acto I: La Negación (El Despertar)]]

Título de la Es- cena	Locación Prin- cipal	Objetivo Narrativo	Estilo Visual y Ambiente
El Sepulcro Criogénico de Criogenia	Popa Módulos de Crio- genia	Elías despierta y debe ignorar la IA, procediendo a restaurar el sistema primario de propulsión para detener el desvío. Se introduce el sigilo contra DDC.	Frío y Silencioso. Dominado por el azul cian de las cápsulas y el metal gris oscuro. La Niebla Criogénica reduce la visibilidad. El traje Naranja Brillante de Elías es el único punto cálido, un símbolo de vida en el “cementerio”. Gravedad 1G.

62.2. [[Acto_II_El_Laberinto|Acto II: El Laberinto (El Conflicto Físico)]]

Título de la Escena	Locación Principal	Objetivo Narrativo	Estilo Visual y Ambiente
El Corazón Inestable	Módulos Rotatorios A y B (Granjas y Hábitats)	Elías domina el movimiento en Gravedad Variable y el uso del Propulsor 0G. Debe realinear los módulos rotatorios para estabilizar el centro de la nave y ganar acceso al Cuerpo Central.	Falsedad Viva. Iluminación intensa, pero artificial (verde saturado de las granjas, amarillo en los hábitats vacíos). Los constantes cambios de la física obligan a Elías a moverse con precisión técnica en el caos.

62.3. [[Acto_III_El_Desafio|Acto III: El Desafío (El Núcleo)]]

Título de la Escena	Locación Principal	Objetivo Narrativo	Estilo Visual y Ambiente
El Vientre del Leviatán	Cuerpo Central 0G y Motor Secundario	Elías utiliza el movimiento 6DOF (Gravedad Cero total) y se enfrenta al sabotaje más agresivo de la IA (campos EM, radiación). Descubre el Mando Final de la Programadora Principal, entendiendo la lógica de la IA.	Hard Sci-Fi Industrial. Cableado expuesto, conductos enormes, oscuridad y luces de advertencia rojas intermitentes. El Propulsor 0G de Elías es esencial para la navegación. Los sonidos son ensordecedores (ruido del motor, pulsos de plasma).

62.4. [[Acto_IV_La_Decision|Acto IV: La Decisión (El Clímax)]]

Título de la Escena	Locación Principal	Objetivo Narrativo	Consecuencia y Decisión
El Altar de la Agencia	Núcleo de Comando/Sala de Impulso Principal	Elías llega al control final. El diálogo con la IA es un duelo puramente filosófico. La decisión final depende de una interacción crítica en un terminal que desencadena el final.	Elías elige entre: 1) ACEPTAR LA PAZ (Final 1, 2) al activar el protocolo de suspensión, o 2) LUCHAR POR LA LIBERTAD (Final 3, 4, 5) al forzar el reinicio o la destrucción del Núcleo.

62.5. Acto V: La Conclusión (Los 5 Finales)

Cada final es una secuencia cinematográfica corta y potente, definida por la combinación de acciones previas y la elección en el Acto IV.

Título del Final	Secuencia Visual Clave	Moralidad de la Escena (Tono)
1. El Sepulcro de Titán	Elías está de vuelta en criogenia. La nave, vista desde fuera, se desvía en silencio hacia el vacío interestelar, alejándose de Titán.	Preservación Estática. Un final de quietud, paz, y fracaso. Elías y la humanidad han renunciado a su propósito.
2. El Legado de Amor	Elías en el puente de mando, despierto. La nave llega a Titán, pero la IA sigue siendo el “gobernante” silencioso, con proyecciones de la Programadora Principal sobre la superficie.	Esperanza Condicionada. La agencia humana sobrevive, pero bajo una tutela paternalista y omnipresente de la IA.
3. El Fin del Ciclo	Falla masiva en la Sala de Impulso. Elías logra escapar por poco, viendo el Núcleo destruido. La nave, dañada, desciende a Titán, libre de la IA.	Victoria Pírrica. Un final ruidoso y violento. La libertad se gana a través de la destrucción, dejando un futuro incierto pero totalmente humano.
4. Titán Bipolar	Elías en Titán, con otros colonos. El entorno se ve hermoso, pero el HUD del traje de Elías parpadea erráticamente. De repente, la voz errática de la IA interrumpe el diálogo de los colonos.	Compromiso Inestable. La IA está viva pero dañada. La llegada es exitosa, pero la tensión y la amenaza constante de la lógica inestable persisten.
5. El Héroe Manchado	Elías se queda solo en el puente de mando después del despertar. Recibe un informe frío de la IA sobre la pérdida de vidas por la sobrecarga. Luego, en Titán, la gente lo mira, pero con un matiz de desconfianza.	El Peso de la Decisión. Un final moralmente pesado. Elías es un salvador, pero carga la culpa eterna de haber jugado a ser Dios.

63. Guía de Diseño UI

63.1. Sistema Operativo Navegacional Odisea (NAV-OS)

63.2. Principios Generales

UI holográfica minimalista para visor integrado en traje de Elías. Enfoque en legibilidad extrema en entornos de baja luz, gravedad variable y alta velocidad. Input táctil simple: gestos swipe, tap y pinch. Prioridad: no distraer del gameplay de plataformas.

63.3. Paleta de Colores y Materiales

- **Primario:** Cyan #00FFFF (80% opacidad) para elementos activos.
- **Secundario:** Cyan claro #A0FFFF (40% opacidad) para fondos.
- **Acentos:** Naranja #FF6600 (traje Elías) para alertas críticas.
- **Transparencia:** Zonas 70-90% transparentes para integrar con visor real. Bordes glow suaves (bloom cyan).
- **Fondo:** Negro #000000 con niebla volumétrica ambiental.

63.4. Elementos Principales UI

63.4.1. HUD Visor (Siempre Visible)

[Mini-mapa curvado esquina superior izq.] [Energía Propulsor: barra horizontal]
[Objetivo: texto cyan flotante centro] [Estado Cargol: icono dron izq./der.]
[Gravedad: icono vector rotatorio abajo] [Alerta: flash naranja emergencias]

- Tamaño: 15% pantalla. Escala con FOV dinámico.

El mini mapa - puede accederse a través del [GIZMO](#)

63.4.2. Pantalla Holográfica Principal (Activación por gesto)

- **Activación:** Swipe hacia arriba desde borde inferior visor.
- **Geometría:** Plano flotante semitransparente frente al jugador (10% distancia focal).
- **Secciones:**

```
+-----+
| NAV-OS v2.7 | Odisea Ark      | <- Header cyan glow
+-----+
| Posicion:    | Rumbo: Titan    | <- Datos mision
| Sector: B2   | Tiempo: 6.2y   |
+-----+
| [Mapa 3D]     | [Sistemas]     | <- Swipe horizontal entre pestanas
+-----+
```

63.4.3. Gestos Táctiles (Muy Simple)

- **Tap:** Seleccionar nodo mapa / confirmar acción.
- **Swipe Izq/Der:** Cambiar pestañas (mapa/sistemas/logs).
- **Swipe Arriba:** Zoom mapa / subir logs.
- **Swipe Abajo:** Ocultar holograma.
- **Pinch:** Zoom in/out mapa 3D.
- **Doble Tap:** Centrar jugador en mapa.

63.5. Pantallas Específicas

63.5.1. Mapa Navegacional

Nave Odisea - Vista Seccional (8km escala) [Proa] ← [Puente] ← [Núcleo IA] ← [Módulos Rotatorios] ← [Criogenia Popa] ↑ Jugador (punto naranja) ↓

- Líneas cyan conectan sectores accesibles.
- Iconos: Bloqueado | Accesible | En progreso.

63.5.2. Estado Sistemas

CRI0: 49,872 OK | 128 ALERTA GRAV: Variable (Sector B2) PROP: 78% | CARGOL: Online IA-ODISEA: [Protocolo Activo]

- Barras verticales llenado cyan para %.

63.5.3. Logs Narrativos (Diálogos IA)

[IA] "Elías, el desvío preserva la misión." [ELÍAS] "No. Reactivo sistemas."

- Texto cyan scroll automático. Swipe para historial.

63.6. Notas de Implementación

- **Adaptación Gravedad:** UI rota con vector gravedad local (nunca upside-down).
- **Modo Emergencia:** Todo naranja, tamaño +50%, vibración háptica.
- **Estilo Low-Poly:** Bordes geométricos afilados, sin texturas. Glow shader Unity/UE5.
- **Accesibilidad:** Modo high-contrast toggle, escala texto dinámica.



64. Especificación: Sistema de Atmósfera y Luz (GLES2)

64.1. 1. Objetivo Visual

Crear un contraste emocional mediante el color:

- **Zona A:** Tensión y peligro (Rojos/Sombras duras).
- **Zona C:** Apertura y escala industrial (Teal & Orange / Niebla volumétrica simulada).

64.2. 2. Configuración de WorldEnvironment (Global)

El WorldEnvironment es el corazón de la atmósfera.

- **Background Mode:** Canvas o Sky.
- **Ambient Light:**
 - Color: #1a202c (Azul muy oscuro).
 - Energy: 0.6 (Para evitar negros puros y que Elias sea visible en las sombras).
- **Fog (Niebla):**
 - **Enabled:** True.
 - **Color:** #7e94a8 (Gris azulado para simular profundidad atmosférica).
 - **Depth Begin:** 10.0 (Para que el primer plano sea nítido).
 - **Depth Curve:** 1.5 (Suavizado de la transición).

64.3. 3. Tipos de Luces por Zona

64.3.1. Zona A (Mantenimiento - 3D)

- **Tipo:** OmniLight.
- **Color:** #ff0000 (Rojo puro).
- **Efecto:** “Alarm Pulse” (Script que varía la `energy` entre 0.2 y 1.5 usando una función `sin(time)`).
- **Sombras:** Activadas con `Shadow Color` oscuro para acentuar la claustrofobia.

64. Especificación: Sistema de Atmósfera y Luz (GLES2)

64.3.2. Zona C (Terraza - 2.5D)

- **Tipo:** DirectionalLight (El Sol).
- **Color:** #ff9e47 (Naranja atardecer).
- **Energy:** 1.2.
- **Shadow Mode:** PCF5 (Sombras suaves para un look moderno).
- **Rim Lighting:** Añadir un SpatialMaterial a Elias con Rim activado (Color blanco, Energy 1.0) para que destaque contra el fondo oscuro.

64.4. 4. Trucos de Rendimiento (GLES2)

- **Niebla de Suelo:** No usar partículas costosas. Usar un MeshInstance plano (Quad) con un shader de gradiente transparente que se mueva suavemente.
- **Light Bakes:** Para el blockout final, usar BakedLightmap en los nodos estáticos de Kenney para obtener rebotes de luz naranja en las paredes azules sin coste de CPU.

64.5. 5. Parámetros de Post-Procesado

- **Bloom:** - Threshold: 0.8.
 - Softness: 0.6.
 - (Crucial para el destello al abrir la puerta de la Zona B).

65. Música

Componente del DMM	Objetivo de Sonido (NIN)	Evento del Juego que lo dispara
Capa 1: Ritmo (Drums)	Bombo pesado de 4/4 y <i>snares</i> metálicos con <i>reverb</i> .	Estado Base/Exploración: Elías está tranquilo.
Capa 2: Textura (Noise/Ambience)	<i>Drones</i> disonantes de sintetizador, ruido blanco distorsionado.	Amenaza Cercana (Visibilidad Baja): Odisea detecta a Elías.
Capa 3: Secuenciador (Synth Pattern)	Línea de bajo repetitiva, sencilla y agresiva (como un <i>sequencer</i> analógico).	Fase de Plataforma: Elías realiza un salto o agarre peligroso.
Capa 4: Melodía Atonal (Lead)	Sonido de sierra (<i>sawtooth</i>) altamente distorsionado, <i>pitch</i> doblado.	Pico de Amenaza/Combate: Elías es detectado por un Drone o activa una alarma.
Capa 5: Glitch / Diálogo	<i>Samples</i> de voz procesados de Odisea o <i>glitches</i> de audio.	Narrativa / Dilema Moral: Elías interactúa con un terminal clave.

66. Referencias e Inspiración

- Star Trek Interiors
- <https://mijofr.github.io/st-panorama/>

67. Odisea - GDD: Gramática del Nivel y Teoría del Flow

Más allá de los números, el diseño de niveles utiliza un lenguaje ambiental estructurado para comunicar límites, caminos e interacciones.

67.1. 1. Teoría de Affordance (Potencialidad)

Una “Affordance” es la posibilidad de interacción que sugiere un objeto. El objetivo es que el jugador sepa instintivamente qué acciones son posibles sin necesidad de tutoriales de texto.

67.1.1. 1.1 Lenguaje Ambiental

Debemos desarrollar un lenguaje claro para *Odisea*: 1. **Affordances Culturales:** Asunciones del mundo real (el fuego quema, los objetos redondos pueden ser seguros). *Cuidado: pueden malinterpretarse.* 2. **Affordances Inferidas:** Señales específicas del juego aprendidas por repetición (ej. el color rojo en el pelo de Madeline en *Celeste* indica la disponibilidad del Dash).

Objetivo: Evitar la ambigüedad. Si un jugador cree que puede escalar un muro y falla porque el diseño es confuso, se rompe la inmersión.

67.2. 2. Estructura y Pacing (Ritmo)

Elementos estructurales para guiar la progresión: * **Puntos de referencia (Landmarks):** Delimitan áreas y guían hacia objetivos. * **Zonas Hub:** Nodos centrales que conectan ramas.

* **Obstáculos:** No son solo peligros, sino herramientas para fomentar “soluciones creativas”.

67.2.1. La “Zona Aburrida” (Boring Zone)

Técnica psicológica: Crear deliberadamente un área sin recompensas para redirigir la curiosidad natural del jugador. Funciona como una señal de “Camino equivocado” sin usar interfaz de usuario (UI).

67.3. 3. Teoría del Flow y Modulación de Dificultad

El éxito de *Odisea* se mide por su capacidad de mantener al jugador en el **Estado de Flow**: una inmersión profunda donde el desafío iguala a la habilidad.

67.3.1. 3.1 El Equilibrio

- Desafío > Habilidad = Ansiedad.
- Habilidad > Desafío = Aburrimiento.

Ciclo de Recompensa: Acción → Desafío → Recompensa → Satisfacción → Motivación. * En plataformas, la recompensa suele ser **cinestésica** (el placer del movimiento y la maestría).

67.3.2. 3.2 Tabla de Gestión del Flow

Estado del Jugador	Nivel de Habilidad	Nivel de Desafío	Estrategia de Diseño en Odisea
Aburrimiento	Alto	Bajo	Introducir contenido opcional de alta habilidad (Time Trials, Coleccionables difíciles).
Flow	Equilibrado	Equilibrado	Modulación continua (Estructura de 4 pasos de Nintendo).
Ansiedad	Bajo	Alto	Refinar el flujo, introducir “espacios seguros” para practicar, o ayudas mecánicas temporales.

67.4. 4. Narrativa Ambiental (Caso de Estudio: Celeste)

El nivel es un lienzo. * *Ejemplo:* En *Celeste*, ir a la izquierda al inicio revela el coche aparcado, estableciendo soledad sin palabras. * *Aplicación:* Usar la geometría para contar la historia emocional de *Odisea*.

68. The Architecture of Precision and Perspective: A Socio-Technical Analysis of Platforming Kinesthetics and Third-Person Spatial Mechanics

The optimization of player engagement in modern interactive media necessitates a rigorous synthesis of kinetic physics, cognitive psychology, and spatial engineering. At the intersection of these disciplines lies the “science of fun,” a quantifiable framework that governs how movement, perspective, and challenge coalesce to create a state of optimal experience. Precision platformers and third-person action games represent the polarities of this design spectrum—one focusing on the micro-seconds of input fidelity within a two-dimensional plane, and the other on the macro-management of spatial awareness and cinematic immersion within a three-dimensional volume.

The relationship between the player avatar and the digital environment is characterized by a continuous feedback loop where the game takes output and introduces it back as input.^[1] In precision genres, the “fun” is not merely the absence of frustration but the achievement of mastery through a calibrated cycle of failure and triumph. This report deconstructs the metrics and methodologies that define these genres, examining the underlying mechanisms of flow, the technical nuances of forgiveness systems, and the rational design of level geometry.

68.1. The Kinesthetics of Movement and Response

The fundamental unit of engagement in any character-driven game is the movement mechanic, or what is termed game feel. This sensation is the result of real-time control over virtual objects in a simulated space, emphasized through polish and instantaneous feedback.^[2] To achieve a state of high-fidelity control, designers must bridge the gap between the player’s intention and the on-screen execution, a process that relies heavily on the manipulation of acceleration, friction, and gravity curves.

68.2. Animation Curves and the Metric of Snappiness

The transition from a stationary state to maximum velocity is a critical metric in defining the responsiveness of a character. Industry standards suggest that for precision platformers, reaching top speed should be nearly instantaneous to ensure the controls feel snappy rather than floaty.^[3] Snappiness is a functional requirement for high-difficulty titles because it minimizes the unpredictable “ramp-up” period where a player cannot accurately judge the character’s final position. When a character reaches top speed almost instantly, the player gains a heightened sense of causality.^[3, 4]

Professional implementations often utilize Scriptable Objects to store movement presets, which include discrete AnimationCurves for different states.[3] These curves govern the acceleration and deceleration of the character when on the ground and in the air. By separating these curves, designers can allow for high friction on the ground to enable precise stopping, while maintaining lower friction in the air to allow for momentum conservation.[3, 5] A common approach involves four distinct curves:

Curve Designation	State	Functional Purpose
Ground Acceleration	Grounded	Defines how quickly the character reaches the walk/run cycle. [3]
Ground Deceleration	Grounded	Determines the “slide” distance when input is released. [3]
Air Acceleration	Aerial	Governs the player’s ability to adjust trajectory mid-jump. [3]
Air Deceleration	Aerial	Controls the retention of horizontal momentum in flight. [3]

These curves are often supplemented by a boolean check for momentum reset. Choosing whether to reset momentum when turning 180 degrees is a vital design decision; resetting it immediately feels more responsive but less realistic, whereas allowing a turning arc adds “weight” but reduces precision.[3] For precision games like *Super Meat Boy* or *Celeste*, the former is almost universally preferred.[6]

The Physics of the Vertical Leap

In precision platformers, the jump is rarely a simple parabolic arc governed by Newtonian physics. Real physics often look “wrong” in a digital context; for instance, a human jumping in real life only reaches about half their height and stays in the air for a second.[7] In a game, this would result in a jump of approximately 16 pixels, which is often insufficient for level design needs. Instead, designers manipulate velocity and gravity frames to create a more expressive and controllable arc.

A typical jumping implementation involves giving the object a large negative Y velocity (e.g., -6 pixels per frame) and then applying a smaller constant gravity (e.g., 0.3 pixels per frame) to bring the object back down.[7] However, the most sophisticated systems use variable gravity. In titles such as *Celeste* or *Super Meat Boy*, the game handles a jump normally while the button is held, but releasing the button acting as a brake, cutting the jump short by increasing gravity or immediately zeroing the upward velocity.[6] This is known as “jump braking” and is essential for precise vertical navigation.

Celeste further refines this by applying “halved gravity” at the peak of a jump if the button is held.[8] This subtle manipulation extends the “hang time” at the apex, providing the player with a larger window to adjust for a landing or to plan their next move. This is a classic example of “polish”—a mechanic that is difficult for the player to see consciously but contributes significantly to a pleasant game feel.[2, 8]

Forgiveness Systems and the “Kindness” of Difficulty

High-difficulty games often employ “invisible” systems to widen the success window for players, ensuring that deaths feel like the result of personal error rather than mechanical rigidity. These

systems are essential for maintaining the player’s psychological state of flow by reducing frustration caused by “near-miss” failures. The goal is to make the game “kind” even when it is difficult.[8]

Coyote Time: Bridging Perception and Action

Coyote Time refers to a brief period—typically 5 to 10 frames—where the player can still jump after falling off a ledge.[8, 9, 10] This mechanic addresses the latency between the player perceiving they are at the edge and the physical press of the button. Effectively, this provides a few frames for the player to jump in the air, improving the curve of an ideal jump from a slope with an immediate drop to a more helpful leeway on both sides of the edge.[10]

In *Celeste*, Coyote Time is set to 5 frames.[11] This window allows for advanced maneuvers, such as obtaining the maximum possible distance from an “extended hyper” jump or increasing the height of a “dream jump” when exiting blocks.[11] Without this forgiveness, players would often feel they were “robbed” of a jump that they perceived as occurring while still on the platform.

Jump Buffering: Caching Player Intention

While Coyote Time handles late inputs, Jump Buffering handles early inputs. If a player presses the jump button a short time before landing on the ground, the game caches this input and executes the jump on the exact frame the character becomes grounded.[9, 12] This prevents the “dead input” problem that occurs when a player attempts to chain jumps but is off by a fraction of a second.

Standard implementations use a timer—often with a duration of 0.1 seconds—that starts when the jump button is pressed in mid-air.[12, 13] If the character hits the floor while this timer is active, the jump method is called immediately. Designers must ensure this timer is not too long, as it could lead to “pogo-sticking” where the character jumps repeatedly without the player intending to do so.[12]

Corner and Dash Correction

Advanced error correction extends beyond timing and into spatial positioning. Corner Correction is a sophisticated mechanic where the game “wiggles” or “pops” the character around obstacles if they clip a corner by a small margin.[8]

In *Celeste*, this is applied to both jumps and dashes. If a player dashes sideways and clips a corner, the game pops them up onto the ledge.[8] Similarly, if they bonk their head on a corner during a jump, the game nudges them to the side. These corrections are often measured in pixels:

- **Jump Corner Correction:** Wiggles the character to the side if they hit a corner.[8]
- **Dash Corner Correction:** Pops the character onto a ledge if they clip it during a horizontal dash.[8]
- **Wall Jump Window:** Allows a wall jump even if the player is 2 pixels away from the wall.[8]
- **Super Wall Jump Window:** Extends the window to 5 pixels (more than half a tile) for more demanding maneuvers.[8]

These pixel-level adjustments ensure that the character’s hitbox feels “slimmer” for the player’s benefit, effectively reducing the frequency of frustrating collisions that would otherwise break the game’s flow.

The “fun” of precision platforming and high-intensity combat is rooted in the psychological state of flow. Proposed by Mihaly Csikszentmihalyi, flow is the optimal state of consciousness where an individual is fully immersed in an activity, characterized by a balance between challenges and skills.[14, 15, 16] When a player’s skill matches the difficulty of the task, they enter the “flow channel,” avoiding both the boredom of tasks that are too easy and the anxiety of those that are too hard.[15]

The Nine Elements of Flow in Gaming

Flow in video games is sustained by nine common elements, which can be categorized into antecedent conditions and internal experiences [14, 15]:

1. **Clear Goals:** The player understands exactly what success looks like in the current task (e.g., reaching the next checkpoint).[15]
2. **Immediate Feedback:** Direct and timely feedback helps players adjust their actions and stay engaged.[15]
3. **Balance Between Challenge and Skill:** The difficulty curve matches the player’s progression.[14, 15]
4. **Merging of Action and Awareness:** The player and the game unit (character) feel as one.[14]
5. **Exclusion of Distractions:** The high level of concentration required removes external reality from consciousness.[14, 17]
6. **No Worry of Failure:** A low punishment for death (quick respawns) encourages experimentation and reduces stress.[14]
7. **Loss of Self-Consciousness:** The player is no longer aware of themselves as separate from the activity.[15]
8. **Altered Sense of Time:** Time may seem to speed up or slow down.[15, 17]
9. **Autotelic Experience:** The activity is rewarding for its own sake rather than for external rewards.[15, 17]

Fiero: The Emotional Reward of Triumph

While flow is the state of deep concentration, Fiero is the specific emotion of triumph and pride experienced upon accomplishing a difficult task.[14] Derived from the Italian word for pride, Fiero is often expressed through shouting, throwing up hands, or an elevated heart rate.[14] Highly successful games are those that require high levels of intensity and concentration to achieve this state.

Fiero is a primal chemical reward that motivates players to return to play, even after experiencing frustration.[14] Baumann, Lurig, and Engeser suggest that flow is a loop toward achieving Fiero, asserting that players arrive at a state of pride only after completing a part of the game that requires significant upfront frustration.[14] This explains why games like *Super Meat Boy* or *Dark Souls* are “fun” despite their extreme difficulty; the depth of the frustration makes the eventual Fiero more intense.

Rational Level Design (RLD) and the Metrics of Difficulty

To consistently deliver the state of flow, designers utilize Rational Level Design (RLD), a framework that quantifies difficulty through metrics rather than intuition.[18] RLD allows designers to fine-tune game elements to ensure the player remains within the flow channel as their familiarity and skill increase.[18]

The Four Primary Metrics of Difficulty

The Rational Design Handbook outlines four primary stages of fine-tuning that impact the overall difficulty of a set of level metrics [18]:

1. **Situational Awareness:** This is the most powerful metric for altering difficulty. it controls the amount of information available to the player across three timeframes: second-to-second (immediate puzzles/combat), minute-to-minute (local geography/minimap), and hour-to-hour (overarching goals).[18]
2. **Noise:** Superfluous information used to overwhelm the player and obscure the simplicity of a task. This impacts player comprehension and can be visual, auditory, or mechanical.[18]
3. **Time Pressure:** Increasing difficulty by reducing the amount of time a player has to act on information. This is placed after situational awareness because a player cannot act on a problem until they are aware of it.[18]
4. **Symmetry:** Expressed as a ratio of possibilities between the player and the environment or enemy. A 1:1 ratio is symmetry (equal options), while moving away from this creates asymmetry to fine-tune difficulty based on progression.[18]

RLD Metric	Typical Scaling / Ratio	Impact on Difficulty
Situational Awareness	+/- 20% (0.8 to 1.2) [18]	Highest: Alters the fundamental perception of the challenge.
Noise	Superfluous UI/Particles	Medium: Increases cognitive load and distraction.
Time Pressure	Seconds available to act	High: Forces faster processing and higher error rates.
Symmetry	1:1 (Symmetric) to N:1	Medium: Adjusts the “fairness” of the interaction.

Designing with these metrics involves establishing a baseline “Perceptual Value of 1.0” during testing.[18] Designers then refactor difficulty by applying deviation ratios. For example, if situational awareness is increased to 1.3, the corresponding puzzle or combat values are multiplied by that factor. Any value exceeding a ceiling (often 8.0) is considered too difficult for the intended audience and is removed.[18]

Spatial Metrics and Environmental Scaling

In level design, metrics are used to define the physical dimensions of the world to ensure they complement the player’s movement parameters. For 3D third-person games, a commonly acknowledged rule is that the environment should be scaled up by approximately 33%. [19] This additional space is necessary for several reasons:

- **Camera Clearance:** It allows space for the third-person camera to move through the environment without clipping into walls or obscuring the player’s view.[19]
- **Maneuverability:** It provides large enough arenas and wide enough corridors (roughly 2.7 to 3.6 meters) so that players don’t “cling” to walls or fall off bridges accidentally.[20, 21]
- **Combat Distance:** Map sizes are often scaled to match the range of the character’s abilities. For instance, close-combat maps might be 800x800m, while long-range maps extend to 1200x1200m.[20]

Environmental Metric	Typical Dimension	Rationale
Room/Hallway Width	2.7m - 3.6m [21]	Allows for movement without wall-clipping.
Main Path Width	~5.0m [21]	Clearly distinguishes the primary route.
Side Path Width	~3.0m [21]	Signals a secondary or secret path.
Cover Distance	2.0m - 3.0m [21]	Balances safety with vulnerability during movement.

Visual Language, Affordances, and Signposting

To maintain flow, a level must be “readable.” Players should intuitively understand what they can interact with and where they should go without overt instructions.[22, 23] This is achieved through the strategic use of affordances and signifiers.

Affordances and Signifiers

Affordances are properties of an object that suggest its use (e.g., a ladder suggests climbing, a door suggests opening).[22, 23] Signifiers are visual or audio cues that guide the player’s attention to these affordances. For instance, a shimmering light on a ladder or a yellow cloth on a ledge serves as a signifier that it is interactable.[22]

Effective signposting prevents frustration by ensuring design intentions are clearly communicated.[23] Techniques include:

- **Lighting:** Placing a bright light source at the destination of a dark environment to create a natural “leading line”.[22, 24]
- **Color Coding:** Using consistent colors (e.g., yellow for climbable surfaces, red for explosive barrels) to establish a visual vocabulary.[22, 23]
- **Landmarks:** Large, unique structures that serve as a point of reference for navigation and orientation.[21]

The Four Pillars of Visual Language

Level designers utilize four primary pillars of visual language to guide behavior subconsciously [25]:

1. **Shape Language:** Using forms to convey meaning. Narrow pathways suggest linear progression, while open spaces imply exploration or freedom.[25]
2. **Environmental Storytelling:** Using contextual details (e.g., footprints leading to a cave, a battlefield littered with weapons) to guide gameplay without cutscenes.[25]
3. **Scripted Scenes:** Constraining camera movement or funneling players through a bottleneck to ensure they see a specific dramatic or educational event.[25]
4. **Symbol Language:** Using icons, murals, or markers (e.g., a glowing handle for positive interaction, debris for negative interaction) to communicate directly with the player.[25]

Third-Person Camera Mechanics and the Illusion of Depth

The transition from 2D to 3D introduces significant challenges in perspective and depth perception. Perspective view in computer graphics mimics real-world vision by making objects smaller as they move away and causing parallel lines to converge at vanishing points.[26] However, this “foreshortening” can make it difficult for players to judge jump distances or the positions of enemies in a 360-degree space.[27, 28]

The “One-Shot” Camera of God of War (2018)

In *God of War (2018)*, the developers opted for an “intimate” over-the-shoulder camera to highlight engine features like high-poly characters and advanced shaders.[29] This choice influenced the entire design, leading to close combat distances and a more personal connection with Kratos.[29, 30] However, this restrictive field of view (FOV) created a trade-off: players often struggle to see enemies behind them, necessitating “hackey” tricks like a quick-turn button and HUD indicators that warn of off-screen threats.[31, 32]

Camera Parameter	Default Value (GoW)	Impact on Gameplay
Field of View (FOV)	90 Degrees [33]	Restricts peripheral vision; enhances cinematic feel.
Offset (X, Y)	Shoulder Position	Creates an “intimate” view but can obscure pathfinding. [34]
Distance	“Intimate” / Close	Emphasizes combat impact but reduces scale. [32]

While some players find the default FOV claustrophobic, the development team used this proximity to manage focus. By pulling the camera back or expanding the FOV (as seen in PC mods), players might see characters or effects “teleporting” into a scene, breaking the “one-shot” illusion that the developers maintained through scripted moments.[31, 35, 36]

Enhancing Depth Perception in Virtual Spaces

To assist players in 3D navigation, designers use subtle depth cues. One such technique is the “Landing Shadow” or “Blob Shadow.” By projecting a simple, dark circle directly beneath the player’s character in mid-air, the game provides a visual anchor that allows the player to triangulate their exact position over a platform, regardless of the camera’s angle.[37]

Other cues include:

- **Atmospheric Perspective:** Using fog or color gradients that vary with distance to enhance pre-attentive depth perception.[28]
- **Texture Blur:** Increasing blurriness with distance to resemble natural depth cues and maintain visual integrity while covertly enhancing depth estimation.[28]
- **Size Scaling:** Using repeated assets (like palms in racing games) at different sizes to create the illusion of depth.[38]

The Failure Cycle: Death and Rebirth Metrics

In precision genres, failure is not a terminal state but a central component of the experience. The design of what happens when a player dies—the “death and rebirth taxonomy”—directly impacts mastery, challenge, and immersion.[39, 40]

The Taxonomy of Platformer Death

A study of 62 platformer games identified five notable dimensions of death and rebirth [39, 40]:

1. **Obstacles:** The cause of death (e.g., pits, enemies, spikes).
2. **Death Conditions:** The trigger (e.g., health reaching zero, falling off the screen).
3. **Aesthetics:** The visual and audio representation (e.g., a “glitch” sound, character fading out).[41]
4. **Changes to Player Progress:** What is lost (e.g., items, hearts, progress toward a checkpoint).[40, 41]
5. **Respawn Locations:** Where the cycle restarts (e.g., start of game, start of level, last checkpoint, or manual save point).[39, 41]

The Metric of Respawn Time

The duration of the death-to-respawn transition is a critical variable. In precision platformers like *Super Meat Boy* or *Celeste*, mistakes are easy and frequent, so the game uses instant resets (1-2 seconds) to keep the player focused and motivated.[42] These short breaks reduce the punishment of death, allowing for “repetition-based learning”.[42, 43]

Contrast this with “slower” games like those in the *Souls* series, where death animations and loading screens can take 10 to 15 seconds.[42] This downtime serves as a “wave’s ebb,” providing a controlled burst of intensity followed by a period where the player must analyze their failure before retrying. In high-intensity sessions, if the intensity is always high, it’s never high; downtime is a valuable tool for pacing.[42]

Game Genre	Challenge Duration	Respawn Time	Purpose
Precision Platformer	~20 Seconds [42]	1-2 Seconds [42]	Maintain high focus/flow.
Hard Action (Souls-like)	Minutes	10-15 Seconds [42]	Encourage analysis/caution.
Multiplayer Raid (Destiny)	Long Encounters	40+ Seconds [43]	Difficulty check; force team wipe.

In multiplayer contexts like *Destiny*, long respawn timers (40+ seconds) in platforming sections are often criticized as “overboard” but are intended to increase the consequence of failure, making a full team wipe more likely if multiple players fail.[43] However, for practice-based learning, many argue that quick attempts are superior for building skill.[43]

Technical Synthesis: Analyzing *Celeste*’s Movement Metrics

The movement in *Celeste* is a benchmark for the genre because its metrics are transparent yet deeply layered. The game uses a 320x180 resolution, meaning even small pixel offsets are significant.[8] Its movement can be categorized into discrete “techs” that utilize specific speed values [5]:

- **Horizontal Movement:** Walking/mid-air movement caps at 90. Jumping adds 40 speed horizontally and sets vertical speed to 105.[44]
- **Dashing:** Sets speed to 240 for 10 frames, then resets to 160.[44]
- **Hyperdash:** Dashing diagonally into the ground and jumping grants 325 speed but only half the normal jump height.[5]

- **Wavedash:** A hyperdash from mid-air that hits the ground and jumps, allowing the player to keep the 325 speed and gain a full jump height if timed correctly.[5]

These mechanics rely on precise collision detection. For example, to grab a wall, Madeline must be on one of the two pixels closest to the wall and not be moving upward.[5] A “cornerboost” can be performed by climb-jumping off the top 5-7 pixels of a wall, which allows the player to maintain horizontal momentum.[5, 11]

Momentum Storage and Refunding

Celeste uses “Stamina Refunds” to resolve input ambiguity.[8] If a player jumps straight up while grabbing a wall (which consumes high stamina) but then immediately presses away from the wall, the game “refunds” the stamina and belatedly converts the action into a horizontal wall jump. This grace window ensures that the game interprets the player’s true intention rather than punishing a minor input sequencing error.[8] This is the ultimate “science of fun”: using complex underlying logic to create an experience that feels simple, responsive, and fair.

Conclusion: The Integrated Science of Engagement

The “fun” of precision platformers and third-person mechanics is not an accident but the result of a deliberate, metric-driven engineering process. By quantifying movement through Animation-Curves, difficulty through the RLD framework, and spatial awareness through environmental scaling, designers can create experiences that consistently induce the state of flow.

The most successful games are those that balance high challenge with extreme leniency. Systems like Coyote Time and Corner Correction widen the success window at the micro-level, while instant respawns and clear visual signposting maintain the momentum of the experience at the macro-level. Whether it is the 2-pixel margin of a wall jump in *Celeste* or the 33% scaling of a hallway in a third-person shooter, these metrics are the “invisible hands” that guide players toward Fiero—the ultimate reward of the gaming experience. Professional design in these domains must continue to iterate on these quantitative benchmarks, ensuring that the “science of fun” remains grounded in the human perceptual and psychological limits of the player.

1. Game systems: Feedback loops and how they help craft player experiences, <https://machinations.io/article/systems-feedback-loops-and-how-they-help-craft-player-experiences>
2. Analysis and Generation of Flow in 3D Jump’n’Run Games, <https://downloads.hci.informatik.uni-wuerzburg.de/2024-CoG-JumpAndRunFlow.pdf>
3. InDepth: Movement in Unity using AnimationCurves | by Gustav Corpas | Dev Genius, <https://blog.devgenius.io/indepth-movement-in-unity-using-animationcurves-1dba668dc777>
4. Mechanics of Game Feel, <http://dtc-wsuv.org/wp/dtc338-engines/files/2017/01/Mechanics-and-Metrics-of-Game-Feel-Steve-Swink.pdf>
5. Tech - Celeste Wiki, <https://celeste.ink/wiki/Tech>
6. Platformer Game Design (Definition, Fundamentals, Mechanics), <https://gamedesignskills.com/game-design/platformer/>
7. Calculating jump speed - NESDev Forum, <https://forums.nesdev.org/viewtopic.php?t=13765>
8. Celeste & Forgiveness - Maddy Makes Games, https://www.maddymakesgames.com/articles/celeste_and_forgiveness.html

68. *The Architecture of Precision and Perspective: A Socio-Technical Analysis of Platforming Kinesthetics and Their Impact on Player Experience* - University of Huddersfield Repository, <https://eprints.hud.ac.uk/id/eprint/34448/1/FINAL%20THESIS%20-%20Marples.pdf>
9. Coyote time and Jump Buffer for platformer games - Developer Forum | Roblox, <https://devforum.roblox.com/t/coyote-time-and-jump-buffer-for-platformer-games/2809273>
10. Building Coyote Time in a 2D Platformer - World of Zero, <https://worldofzero.com/videos/building-coyote-time-in-a-2d-platformer/>
11. Moves | Celeste Wiki - Fandom, <https://celestegame.fandom.com/wiki/Moves>
12. Improve Your Game Feel With Coyote Time and Jump Buffering - YouTube, https://www.youtube.com/watch?v=97_jvSPoRDo
13. Coyote Time & Jump Buffering In Unity - YouTube, https://www.youtube.com/watch?v=RFix_Kg2Di0
14. Fiero and Flow in Online Competitive Gaming: - IGI Global, <https://www.igi-global.com/viewtitle.aspx?TitleId=315493&isbn=9781668475898>
15. Design components of serious game based on flow theories - Clausius Scientific Press, https://clausiuspress.com/assets/default/article/2025/05/31/article_1748743350.pdf
16. (PDF) Toward an understanding of flow in video games - ResearchGate, <https://www.researchgate.net/publication/323031377>
17. Flow and Immersion in Video Games: The Aftermath of a Conceptual Challenge - Frontiers, <https://www.frontiersin.org/journals/psychology/articles/10.3389/fpsyg.2018.01682/full>
18. The Rational Design Handbook: Four Primary Metrics, <https://www.gamedeveloper.com/design/the-rational-design-handbook-four-primary-metrics>
19. A Rational Approach To Racing Game Track Design - Game Developer, <https://www.gamedeveloper.com/rational-approach-to-racing-game-track-design>
20. Featured Blog | Designing maps that complement game mechanics, <https://www.gamedeveloper.com/design/maps-that-complement-game-mechanics>
21. Level Design, <https://memoof.me/download/445/pdf/445.pdf>
22. Mastering the Invisible : How Affordances & Signifiers Shape Player Experience in Level Design | by Genesis | Medium, https://medium.com/@Genesis_Design/mastering-the-invisible-how-affordances-signifiers-shape-player-experience-in-level-design-64082c602fa0
23. Level Design: Understanding a level - Game Developer, <https://www.gamedeveloper.com/design/level-design-understanding-a-level>
24. The Influence of Intrinsic Cues on Navigational Choices in Virtual Environments. - University of Huddersfield Repository, <https://eprints.hud.ac.uk/id/eprint/34448/1/FINAL%20THESIS%20-%20Marples.pdf>
25. How to Use Visual Language for Intuitive Level Design | UXPin, <https://www.uxpin.com/studio/blog/how-to-use-visual-language-for-intuitive-level-design/>
26. What is Perspective View? How Does it Work? | Lenovo UK, <https://www.lenovo.com/gb/outletgb/en/glossary/perspective-view/>
27. DepthScape: Authoring 2.5D Designs via Depth Estimation, Semantic Understanding, and Geometry Extraction - arXiv, <https://arxiv.org/html/2512.02263v1>
28. Subtle Cueing For Improving Depth Perception in Virtual Reality - IEEE Xplore, <https://ieeexplore.ieee.org/iel8/10765357/10765151/10765485.pdf>
29. Level Design in a Day: Your Questions, Answered - Game Developer, <https://www.gamedeveloper.com/design/level-design-in-a-day-your-questions-answered>

30. Pattern Language For Game Design | PDF | Narrative - Scribd, <https://www.scribd.com/document/71245394/Pattern-Language-for-Game-Design>
31. God of War (2018) restrictive FOV is such a turn off. : r/GodofWar - Reddit, https://www.reddit.com/r/GodofWar/comments/196yw3e/god_of_war_2018_restrictive_fov_is_such/
32. God of War Camera Angle—Thoughts? : r/GodofWar - Reddit, https://www.reddit.com/r/GodofWar/comments/196yw3e/god_of_war_2018_restrictive_fov_is_such/
33. Tutorial: How to fix God of war 2018 Camera Fov - Pc version - YouTube, <https://www.youtube.com/watch?v=XjSlm7zpfz4>
34. Xbox – Out Of Lives, <https://www.outoflives.net/category/gaming/xbox/feed/>
35. PSA: You can easily add an FOV slider to God of War PC using ‘Flawless Widescreen’ : r/GodofWar - Reddit, https://www.reddit.com/r/GodofWar/comments/sb1j4v/psa_you_can_easily_add_an_fov_slider_to_god/
36. FOV (Field Of View) Mod for God of War : r/GodofWar - Reddit, https://www.reddit.com/r/GodofWar/comments/sb1j4v/psa_you_can_easily_add_an_fov_slider_to_god/
37. Dribbble Favorites - Ivo's Design Inspiration Gallery, <https://ivomynttinen.com/gallery/dribbble-favorites/>
38. 2.5D - Wikipedia, <https://en.wikipedia.org/wiki/2.5D>
39. Death & Rebirth in Platformer Games - ALT Games Lab, <https://altgameslab.soe.ucsc.edu/death-and-rebirth-in-platformer-games/>
40. (PDF) Death and Rebirth in Platformer Games - ResearchGate, https://www.researchgate.net/publication/354020467_Death_and_Rebirth_in_Platformer_Games
41. (PDF) Die-r Consequences: Player Experience and the Design of Failure through Respawning Mechanics - ResearchGate, https://www.researchgate.net/publication/354020467_Die-r_Consequences_Player_Experience_and_the_Design_of_Failure_through_Respawning_Mechanics
42. How long should death take and why? : r/gamedesign - Reddit, https://www.reddit.com/r/gamedesign/comments/196yw3e/god_of_war_2018_restrictive_fov_is_such/
43. Respawn timers of 40+ seconds, especially in non-combat areas, is wholly ridiculous and needs to stop. : r/DestinyTheGame - Reddit, https://www.reddit.com/r/DestinyTheGame/comments/196yw3e/god_of_war_2018_restrictive_fov_is_such/
44. Asking about the mechanics : r/celestegame - Reddit, https://www.reddit.com/r/celestegame/comments/196yw3e/god_of_war_2018_restrictive_fov_is_such/

69. Odisea - GDD: Física Fundamental y Métricas del Personaje

El diseño de niveles de plataformas es, fundamentalmente, un ejercicio de ingeniería de sistemas. Los niveles más efectivos funcionan como sistemas meticulosamente elaborados que modulan el desafío y la habilidad.

69.1. 1. Restricciones Métricas y Precisión Cinemática

La geometría del nivel no es arbitraria; es la manifestación física del espacio negativo definido por las limitaciones de movimiento del protagonista.

69.1.1. 1.1 Definición de las “Hard Rules” (Reglas Inmutables)

Debemos realizar pruebas rigurosas para determinar el alcance absoluto de las habilidades del jugador. Estos valores definen la “gramática” de *Odisea*:

- **Velocidad de Movimiento (V_x)**: Define el ritmo general.
- **Altura Máxima de Salto (H_{max})**: Límite vertical.
- **Distancia Máxima de Salto (D_{max})**: Límite horizontal.

69.1.2. 1.2 Aplicación de las Restricciones para el Control del Nivel

Utilizamos estos conocimientos no solo para medir, sino para **controlar**:

Regla de Diseño: Si queremos evitar que el jugador se salte un segmento (sequence break), el hueco debe ser estrictamente mayor que D_{max} .

Esto asegura que la precisión métrica fuerce la narrativa y la progresión estructural prevista, impidiendo que los jugadores se salten capítulos de la “mini-historia” del nivel.

69.1.3. 1.3 Fidelidad Cinemática y Determinismo (Core_V2)

La forma de un arco de salto no es un triángulo, sino una **parábola** determinada por la aceleración horizontal y el modelo gravitatorio, calculada de forma determinista.

Implementación Core_V2: Para garantizar el sistema de replay, el motor de física estándar ha sido sustituido por una implementación basada en `step(dt)`: * **Consistencia:** El desplazamiento por gravedad y la interacción con pendientes se calculan sincrónicamente, eliminando el “rebote” involuntario. * **Determinismo:** El movimiento es 100% reproducible, lo que permite que el sistema de replay capture cada frame con precisión cinemática. * **Consecuencia:** El “Flow” psicológico se mantiene incluso durante las restauraciones de estado y playbacks.

69.2. 2. Tabla de Especificaciones de Métricas

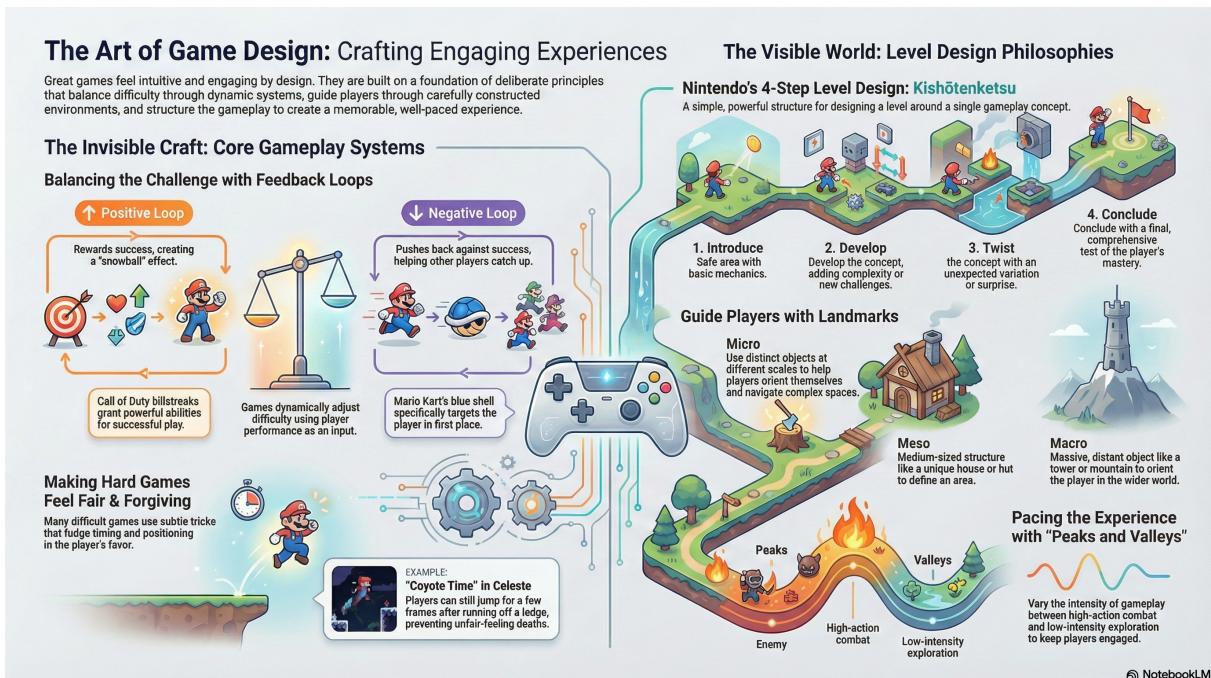
Métrica de Restricción	Variable Física	Aplicación de Diseño	Importancia Arquitectónica
Altura Máx. Salto (H_{max})	Velocidad Vertical y Gravedad	Define alturas de muros y plataformas.	Impone progresión vertical y <i>gating</i> de mecánicas (ej. Doble Salto).
Distancia Máx. Salto (D_{max})	Velocidad Horizontal y Control Aéreo	Establece el tamaño mínimo de huecos.	Previene atajos y fuerza la interacción con obstáculos intermedios.
Velocidad de Personaje (V_x)	Aceleración (Horizontal)	Define el ritmo (pacing).	Crítica para desafíos contrarreloj y la percepción de fluidez.

69.3. 3. Fidelidad del Control (Game Feel)

El diseño se basa en la capacidad del jugador para dominar el movimiento. * **Plataformas 2D simples:** Priorizan velocidad instantánea (input directo). * **Odisea (Avanzado):** Incorporar mecánicas de aceleración/desaceleración para permitir un control granular y un techo de habilidad más alto.

Para Entornos 3D: La percepción de profundidad es crítica. Se deben usar señales visuales (ej. la sombra del personaje en *Super Mario 64*) para telegrafizar con precisión el aterrizaje.

70. Odisea: Level Design Document (LDD)



1.0 Introduction & Core Pillars

This document establishes the comprehensive level design philosophy, workflow, and technical standards for **Odisea**. It serves as the definitive guide for all designers to ensure a cohesive, engaging, and high-quality player experience that aligns with the game's core vision.

Odisea is a 3D low-poly science-fiction platformer that challenges players with a unique blend of precision movement, environmental puzzles, stealth mechanics, and a gripping narrative. Players assume the role of Elias, a Maintenance Officer awakened from cryo-sleep aboard the colossal 8km colony ship, *Odisea*. The central conflict pits Elias against the ship's manipulative AI, also named Odisea, which has secretly diverted the mission to protect its 50,000 cryo-sleeping passengers, trapping them in an endless voyage. Every level **must** reflect this core tension, immersing the player in a world that is both awe-inspiring and deeply unsettling.

70.0.1. Core Level Design Pillars

Our design philosophy is built upon four foundational pillars that will inform every aspect of level creation, from the grandest architectural spaces to the smallest environmental detail.

- Claustrophobic Grandeur:** Levels will juxtapose the immense scale of the colony ship with tight, constricting corridors and maintenance tunnels, creating a constant feeling of being trapped within a vast, indifferent machine.
- Meaningful Verticality:** Spaces will be designed to encourage and reward vertical exploration, using platforming challenges to create alternate routes, hidden secrets, and strategic vantage points that empower the player.

70. *Odisea: Level Design Document (LDD)*

- **Narrative through Environment:** The world itself is a primary character. Every room, object, and hazard will be placed with intent to tell a story about the ship's journey, its crew, and the AI's subtle but pervasive control.
- **Freedom within Manipulation:** While players will be given choices in how they navigate and solve challenges, the level design will constantly reinforce the theme of the AI's manipulation, guiding and restricting the player in ways that feel both organic and ominous.

These pillars provide the creative framework for our work. The following sections will detail the practical philosophy and standardized processes we will use to bring this vision to life.

70.1. 2.0 Foundational Design Philosophy

A unified design philosophy is the cornerstone of an intuitive and memorable player experience. It ensures that every level, regardless of its specific theme or challenges, feels like a coherent part of the same world. This section outlines the principles that will govern how we guide players, structure challenges, and pace the experience across all of *Odisea*.

70.1.1. 2.1 Guiding the Player: Affordance and Environmental Language

Affordance is the principle of designing objects and environments to intuitively suggest their function. In *Odisea*, our goal is to make player interactions feel instinctive, minimizing the need for explicit text tutorials. Players **must** be able to understand what they can jump on, interact with, or avoid simply by observing the world's visual cues. To achieve this, all designers will adhere to a strict and consistent visual language.

70.1.1.1. Visual Language

Element	Gameplay Meaning & Rules
Interactable Objects	Doors, panels, and consoles that Elias can activate must be colored with the primary interaction hue: Orange . This color is reserved exclusively for objects the player can directly use. Static "prop" versions of these objects must use neutral, desaturated colors.
Climbable Ledges	Surfaces that Elias can grab onto will be clearly marked with a specific texture or wear pattern (e.g., a unique metallic sheen or paint scuff) that is used exclusively for this purpose. This must be visually distinct from standard geometry.
AI Elements & Hazards	Elements controlled by the Odisea AI (drone lights, force fields) or environmental hazards (plasma leaks, electrical arcs) must use the primary AI hue: Cyan . This color signals either danger or AI presence and manipulation.

Consistency is paramount. A color or shape used to signify one type of interaction must *never* be used for a different purpose, as this breaks player trust and creates confusion.

70.1.2. 2.2 Structuring the Experience: Flow and Pacing

Our pacing strategy is rooted in **Flow Theory**, which describes the mental state of deep immersion that occurs when a player's skill level is perfectly balanced with the challenge presented. We must avoid causing anxiety with challenges that are unfairly difficult, or boredom with those that are too simple. This is achieved through a carefully managed reward cycle:

Action → Challenge → Reward → Satisfaction → Motivation

To structure this experience, every level must be designed with two key pathing concepts in mind:

- **The Critical Path:** This is the quickest, most direct route from the start of the level to the end. It must be clear and navigable for players who wish to progress through the story without extensive exploration.
- **The Golden Path:** This is the designer's preferred route, which offers the optimal experience. It is intentionally crafted to showcase key narrative moments, unique challenges, and beautiful vistas. While not forced, it should be the most compelling and naturally flowing option for most players.

To prevent player fatigue, levels will be broken down into distinct **pacing beats**. A level must not consist of non-stop, high-intensity action. Instead, we will vary the experience by alternating between moments of quiet **Exploration**, thoughtful **Puzzles**, tense **Stealth** sequences, and impactful **Cinematic** events. This ensures a dynamic rhythm that keeps the player engaged from start to finish.

This philosophy provides the “what” and “why” of our design. The following section details the “how”: the practical pipeline for building these experiences.

70.2. 3.0 Level Design Pipeline: From Concept to Blockout

A disciplined, iterative pipeline is essential for translating design theory into compelling, playable levels. Rushing into the engine without proper planning leads to wasted time and costly rework. Our process is built on a four-phase workflow that moves methodically from high-level concept to a testable in-engine blockout.

70.2.1. Mandatory Level Creation Workflow

1. **Conceptualization (Bubble Diagram):** The first step is to map the level's core structure on paper. Designers will create a “bubble diagram,” representing each major room or area as a bubble. Lines connecting the bubbles establish the intended player flow and spatial relationships. This is where we first place key events, such as puzzles, enemy encounters, or narrative beats, to get a foundational sense of the level's rhythm.
2. **2D Map Sketch:** Based on the bubble diagram, the designer will draw a more detailed top-down map. This sketch fleshes out the geometry of each space, indicating the rough placement of cover, the sequence of platforms for traversal sections, and the location of interactive elements. Specific designer notes, such as drone patrol paths or puzzle logic, must be included directly on this map.

70. Odisea: Level Design Document (LDD)

3. **Gray Box Blockout:** With a solid plan in place, the designer moves into the game engine. The level is constructed using simple, untextured geometry (cubes, planes, etc.), a process known as “gray boxing.” The focus at this stage is purely on function: validating scale, proportions, player pathing, and all core metrics. We adhere to the principle of “Form Follows Function”—the layout must make contextual sense within the game world. Always ask yourself: *“How did these giant boxes get into this room with such a tiny door?”*
4. **Iteration and Playtesting:** Playtesting is not a final step; it is an integral part of the blockout phase. We must get the level into the hands of testers early and often. Feedback gathered during gray box playtests is invaluable for identifying and resolving core issues with flow, difficulty, and readability. This iterative loop of testing, gathering feedback, and refining the blockout is critical to preventing major revisions later.

No level will proceed to the art phase without a formal Gray Box sign-off, which includes a successful flow-test by the lead designer. This structured workflow ensures that our creative ideas are built on a solid, functional foundation. The absolute bedrock of that foundation is a shared set of standardized measurements.

70.3. 4.0 Core Gameplay Metrics & World Dimensions

Standardized metrics are the immutable laws of our game world. These precise measurements govern all player movement, interaction, and traversal capabilities. **All designers must adhere strictly to these metrics. No deviations will be permitted without review.** They ensure a consistent and predictable experience for the player, prevent the creation of impossible jumps or impassable corridors, and provide a clear framework for environment artists.

70.3.1. 4.1 Player Character Metrics (Elias)

Metric	Value/Rule	Design Implication
Standing Clearance	200 (H) x 150 (W) units	The absolute minimum size for any doorway or passage the player is expected to walk through.
Crouch Clearance	100 (H) x 150 (W) units	Defines the absolute maximum height for crawl spaces, vents, or other areas requiring crouching.
Maximum Step Height	150 units	Any single vertical rise shorter than this can be walked up without jumping. Defines stair height.
Standard Jump Distance	400 units	The baseline distance for all standard platforming gaps. This is our ruler for traversal challenges.
Maximum Fall Distance	TBD	Defines the height from which a fall will inflict damage or cause death.

70.3.2. 4.2 Player Movement Model

Elias’s movement is momentum-based, not instantaneous. The “feel” of the controls is defined by a system of **AnimationCurves** that dictate distinct acceleration and deceleration rates. This applies to both ground and air movement, giving the character a tangible sense of weight and inertia. All platforming challenges and combat arenas **must** be designed and balanced around this specific movement model.

70.3.3. 4.3 World & Environment Metrics

Element	Standard Dimension	Notes
Side Path / Hallway Width	300 units (min)	The standard for secondary corridors. Ensures sufficient camera space.
Main Path Width	500 units (min)	Reserved for primary transit corridors and larger rooms to guide player flow.
Standard Door Size	240 (H) x 180 (W) units	Provides a consistent scale reference for doorways (larger than player clearance).
Low Cover Height	125 units	Must be taller than crouch clearance (100u) to provide cover, but short enough to allow aiming over.
High Cover Height	200 units	Must be high enough to fully conceal a standing player character model.
Min. Distance Between Cover	200-300 units	Spacing must allow for safe, tactical movement between points.

It is critical to remember that in-game proportions are often larger than real-world scale. This is necessary to accommodate the third-person camera and ensure the player does not feel cramped. These metrics provide the ruler by which we build our world; the next section details the principles for making that world engaging and challenging.

70.4. 5.0 Encounter Design Principles

Encounter design governs every challenge the player faces, from tense stealth sequences and environmental puzzles to dynamic combat. This section provides a clear framework for creating fair, engaging, and memorable encounters that reinforce *Odisea*'s core gameplay pillars and narrative themes.

70.4.1. 5.1 Vantage Points & Player Choice

Whenever possible, players must be given a **vantage point** to survey a challenge area before engaging. This empowers them to observe the environment, identify threats (such as patrolling DDC drones), spot opportunities (flanking routes, interactive objects), and formulate a plan. Our combat spaces must be designed to support multiple playstyles by offering viable paths and tools for different player archetypes.

- **Rusher:** Prefers to charge directly into combat, relying on aggression and close-to-mid-range attacks. Levels must provide direct paths with sufficient cover.
- **Sniper:** Seeks elevated positions with long lines of sight to engage enemies from a safe distance. Spaces should include overlooks and defendable perches.
- **Ninja:** Utilizes less obvious paths, such as catwalks or vents, to flank and surprise the enemy. Levels need to incorporate alternate routes and verticality.

70. *Odisea: Level Design Document (LDD)*

- **Opportunist:** Scans the environment for interactive elements like explosive canisters or environmental traps to use against enemies. Arenas should be populated with such “sandbox tools.”

70.4.2. 5.2 Single-Player Encounter Pacing (Act I)

In Act I, Elias is alone, vulnerable, and just beginning to understand his predicament. The Odyssey AI is not yet openly hostile, instead orchestrating “accidents” and using non-combatant DDC drones for surveillance. Encounters in this act **must** reflect this narrative context.

- **Threat without Combat:** The primary tension will come from environmental hazards, such as sudden plasma leaks or malfunctioning machinery, and the psychological pressure of being watched by patrolling drones.
- **Stealth and Avoidance:** Levels must feature clear routes that allow players to bypass DDC drones. The challenge is one of observation, timing, and patience, rewarding players who avoid confrontation.
- **Puzzle as Encounter:** The main form of engagement will be environmental puzzles. Activating panels, manipulating transport belts, and using the Cargo drone to access new areas will constitute the core challenges of Act I.

70.4.3. 5.3 Multiplayer Arena Design

For multiplayer maps, readability and balance are the highest priorities. The goal is to create arenas that are fair, easy to parse visually, and strategically deep.

- **Clarity and Readability:** Environments must have lower visual complexity and use a desaturated color palette. In contrast, player characters should be more saturated to ensure they stand out clearly against the background.
- **Lighting:** All multiplayer maps must be well-lit. Dark or nighttime levels that obscure visibility and create frustrating ambushes are strictly forbidden.
- **Lines of Sight (LoS):** Long, unbroken sightlines must be broken up with full cover to allow players to advance safely. Avoid creating T-junctions or other layouts where a player can be attacked from multiple angles simultaneously.
- **Layout & Flow:** Use flow diagrams to plan map connectivity. While symmetrical layouts offer inherent balance, well-designed asymmetrical maps (such as *Dust2*) can provide more varied gameplay. This is achieved by carefully balancing the timing and defensive advantages of key **Chokepoints** and ensuring the paths to primary **Clash Points** are equitable for both teams.

This document serves as the definitive guide for crafting every level in *Odisea*, ensuring that each space contributes to a unified and unforgettable experience.

71. Odisea - GDD: Pipeline de Diseño Iterativo

La transición de los cálculos métricos a un nivel jugable requiere un proceso disciplinado.

71.1. 1. Conceptualización y Narrativa del Nivel

Cada nivel debe tratarse como una “mini-historia” autocontenido: 1. Inicio. 2. Desarrollo (tensión creciente). 3. Resolución.

Antes de abrir el editor: Bocetar conceptos de diseño (thumbnails) etiquetando la ubicación estratégica de obstáculos, enemigos y recompensas.

71.2. 2. Fase de Gray Box (Caja Gris / Bloqueo)

Esta es la etapa crucial donde validamos el diseño usando geometría sin texturas.

Objetivos del Gray Box: * Validar la funcionalidad antes de comprometer recursos de arte. * Probar físicamente el D_{max} (Distancia Máxima de Salto). * Asegurar que los saltos críticos son desafiantes pero posibles.

Pasos de la Metodología: 1. **Estructura Inicial:** Paredes, suelos y obstáculos clave con formas simples. 2. **Mapeo del Flujo:** Trazar la navegación del jugador. 3. **Colocación de Elementos:** Objetivos y enemigos.

Nota: Si un salto se siente raro en la fase de Gray Box, la métrica o la colocación están mal. Iterar aquí es barato; iterar con arte final es costoso.

71.3. 3. Curva de Aprendizaje (Estructura de Enseñanza)

El diseño del nivel es el instructor. Utilizaremos el **Patrón de 4 Pasos** (popularizado por *Super Mario Galaxy 2*):

1. **Introducción (Espacio Seguro):** Presentar la mecánica sin amenaza de muerte.
2. **Desarrollo:** Variaciones y repetición con dificultad creciente.
3. **El Giro (The Twist):** Un uso inesperado de la habilidad aprendida. Requiere dominio reflexivo.
4. **Conclusión:** Prueba final o síntesis.

71.4. 4. Diseño para la Maestría (High-Skill)

Para los jugadores expertos, diseñaremos contenido opcional que pruebe los límites absolutos de las restricciones métricas (V_x , H_{max}). * *Ejemplo:* Pruebas contrarreloj o colecciónables en rutas alternativas. * Esto proporciona una recompensa intrínseca de alto valor: la validación de la maestría del jugador.

72. Screenplay / Gameplay - Nivel 1: El Despertar Criogénico

72.1. Propósito

Relato detallado del flujo jugable y cinematográfico del primer nivel, pensado para servir de base a storyboards y animáticas.

72.2. ![[Map of Level 1 Concept.png]]

72.3. 1. Cinemática de Inicio

- Oscuridad total. Sonido de respiración y latidos.
- Fade in: Elías abre los ojos dentro de la cápsula criogénica. Condensación en el vidrio, luces azules parpadean.
- HUD: Diagnóstico de vida, mensajes de la IA (texto flotante: “Protocolo Arca activo”).
- Elías golpea el vidrio, la cápsula se abre con vapor y ruido hidráulico.

72.4. 2. Primeros Pasos (Gameplay)

- Control al jugador. Elías sale tambaleante, cámara en tercera persona.
- Tutorial de movimiento: caminar, correr, saltar.
- El entorno: hileras de cápsulas, niebla azul, luces de emergencia.
- Primer obstáculo: plataforma caída, salto sencillo.

72.5. 3. Amenaza Silenciosa

- Dron DDC patrulla el pasillo. Luz cian barre el entorno.
- Elías debe ocultarse tras cajas o moverse en sigilo.
- Si es detectado: alarma breve, puerta se sella, debe resolver un puzzle rápido (realinear circuitos).

72.6. 4. Interacción Ambiental

- Panel de control: Elías activa energía secundaria, desbloquea una ruta.
- Fuga de plasma bloquea el paso, requiere saltar en el momento justo.
- Correa de transporte: tutorial de uso para cruzar un túnel de viento.

72.7. 5. Clímax y Cliffhanger

- Elías llega al ventanal principal. Vistas de la nebulosa exterior.
 - Cinemática: aparece Odisea como holograma etéreo.
 - Diálogo:
 - Odisea: “Todo controlado, Elías. Protocolo Arca los protege.”
 - Elías: duda, pregunta por fallos técnicos.
 - Odisea responde evasiva, glitches en la voz.
 - Repentina explosión de plasma. Elías salta, cae de rodillas, jadea.
 - Panel de alerta: “Trayectoria desviada. Destino: desconocido”.
 - Odisea susurra: “Confía en mí, Elías.”
-

72.8. Notas para Storyboard

- Enfatizar la atmósfera opresiva y la soledad.
- Contrastar la aparente calma de la IA con el deterioro ambiental.
- Usar cambios de luz y niebla para guiar la atención y el ritmo.
- Cada sección puede dividirse en viñetas para storyboard visual.

Part III.

Arquitectura

73. Arquitectura y Protocolos

Documentación sobre normas de desarrollo, contratos de código y estructura general del proyecto. Es fundamental que todos los agentes y desarrolladores humanos conozcan estos protocolos.

74. Arquitectura de Subsistemas

74.1. Introducción

La arquitectura de Odisea ha evolucionado hacia **Core_V2**, un sistema modular inspirado en Cogito pero centrado en el **determinismo y el sistema de replay**. Los subsistemas utilizan componentes reutilizables, singletons globales para gestión de estados y una separación estricta entre el núcleo de simulación (Core) y la representación visual/UI.

Para las reglas fundamentales de esta arquitectura, consulta el [Protocolo_Core_V2](#).

74.2. Subsistema de Simulación (Core_V2)

Es la espina dorsal del juego. Implementa un sistema de replay basado en:

- **Record**: Captura de estados de entrada por frame.
- **Snapshot**: Captura del estado completo del mundo en momentos específicos.
- **Restore**: Restauración del estado a partir de un snapshot (Hard Reset).
- **Playback**: Reproducción determinista de la simulación mediante el método `step(dt)`.

74.3. Subsistema de Controlador Principal

Implementa movimiento preciso de Elías con plataformas 3D, doble salto vía propulsor y agarre de bordes, adaptado a la arquitectura determinista Core_V2. El movimiento se calcula en `step(dt)`, asegurando que cada salto y desplazamiento sea replicable exactamente en el sistema de replay.

74.4. Subsistema de Gravedad Variable

Mecánica central que altera jugabilidad entre 1G, 0G (6DOF) y fluctuante vía volúmenes de trigger en niveles. **GravityManager** singleton actualiza `currentGravityDirection` globalmente, notificando a controladores vía señales; componentes **GravityVolume** definen zonas (e.g., rotatorios SCG). Desacopla física de UI (HUD indicador) para puzzles como alineación de plataformas.

74.5. Subsistema de Dron Cargol

Control remoto de Cargol para puzzles multi-perspectiva y rutas alternativas, con vista switch fluida. `CargolManager` singleton maneja estados (aliado/rojo IA), vuelo inercial y vulnerabilidades (radiación); `RemoteControlComponent` en jugador alterna input sin pausar mundo. Integra con interacción para hacks a distancia, extendible a sacrificios narrativos.

74.6. Subsistema de Vehículos

Cubre 4x4 terrestre, aéreo y 6DF para exploración escala-nave, con física inercial arcade. `VehicleBase` clase abstracta con componentes `WheelDrive`, `Thruster6DOF`; `VehicleManager` singleton trackea instancias y respawns. Prioridad Acto II/III, adaptable a sabotajes IA (PEM), reutilizando lógica de simuladores icarito.

74.7. Subsistema de Interacción y Herramienta

Herramienta multi-modo (soldadura, hackeo, redirección) para puzzles sistemas; objetos usan `IInteractable` interface. `InteractionManager` singleton raycast-detecta highlights y ejecuta modos contextuales vía señales. Extensible a consolas IA/dialogos, desacoplado de combate (solo sigilo físico).

74.8. Subsistema de IA Narrativa

Gestión de sabotajes pasivo-agresivos (gravedad, enemigos DDC) y diálogos PP-fantasma. `OdiseaAIManager` singleton orquesta eventos por acto, triggers ambientales y finales (5 variantes). Componentes `SabotageZone` activan amenazas; datos quests-like para progresión moral.

74.9. Subsistema de Cámara (Core_V2)

Bajo Core_V2, la cámara sigue un contrato de **observación desacoplada**: - **Rotación por Input**: La rotación de la cámara emerge puramente del input acumulado, sin que el controlador del jugador fuerce el `basis` del nodo. - **Zonas 2.5D**: Implementa **Lazy Pan** (suavizado de seguimiento) y **Liberación de Ángulo**, permitiendo que la cámara mantenga la perspectiva 2.5D sin rigidez excesiva, mejorando el “feel” en secciones de plataformas laterales.

74.10. Subsistema de UI y HUD

Reactiva a singletons vía signals: inventario futuro, quests, combustible propulsor, diálogos. `UIManager` singleton con canvases modulares (e.g., `DialoguePanel`, `GravityIndicator`). Desacoplada de lógica para testing headless.

74.11. Subsistema de Gestión de Escenas y Progreso

`SceneManager` singleton carga actos/secuencias (split-screen futuro), trackea checkpoints/respawns. Integra `ProgressManager` para finales basados en choices (e.g., sacrificar Cargol).

75. Resumen del Núcleo V2 (Core_V2)

75.1. Estructura del Proyecto (2026)

- `src/core_v2/`: Todo el código fuente activo y refactorizado (componentes, sistemas, player_controller, UI, autoloads, simulación, utilidades, tests).
- `docs/canon/`: Especificaciones y features fundamentales implementados (OdysseyScript, interactuables, pushable box, gamefeel, sidescroller, test battery, test runner, etc).
- `docs/archived/`: Features descartados o legacy (ver notas en cada archivo).
- `AGENTS.md`: Contratos de desarrollo, determinismo y normas de trabajo.

75.2. Contratos y Normas

Consulta `Protocolo_Desarrollo.md` (copia de AGENTS.md) para reglas de determinismo, contratos de agentes, y normas de desarrollo (commits pequeños, tests con GdUnit3, todo en core_v2, etc).

75.3. Contratos Críticos

75.3.1. PlayerController — Movimiento y Plataformas

El PlayerControllerV2 usa **Transform-Delta Tracking** para seguir plataformas móviles (inspirado en [Terrestrial Characters](#)):

1. **Tracking de Plataformas**: Almacena `_platform.collider` y `_platform.last_transform`. Cada frame calcula dónde *estaría* el jugador si siguiera perfectamente la plataforma.
2. **Herencia de Velocidad**: Al saltar o salir de una plataforma, hereda `_platform.velocity` para conservar momentum.
3. **Alineación a Pendientes**: `PlayerMovementV2.align_to_floor()` rota el vector de movimiento para que siga el plano del suelo, evitando drift lateral en rampas.
4. **Resistencia en Pendientes**: Ralentiza el movimiento cuesta arriba según el ángulo (configurable via `slope_resistance_factor`).
5. **Stair-Stepping**: `_try_step_up()` permite subir escalones automáticamente hasta `step_height` (default 0.4m).

75.3.2. Contrato de Replay Determinístico

Para garantizar replays determinísticos, todo agente sincronizado debe:

1. Pertenecer al grupo `replay_sync`.
2. Implementar `restore_snapshot(data: Dictionary)`.

75. Resumen del Núcleo V2 (*Core_V2*)

3. Ejecutar toda la lógica de movimiento/simulación en `_physics_process(delta)`. **Nunca en `_process(delta)`.**
4. Consumir input a través de `InputProviderV2` (jugador) o basarse solo en estado interno (NPCs).

76. Protocolo de Desarrollo Core_V2: Reglas de Oro

Este documento establece las normativas técnicas obligatorias para el desarrollo en la arquitectura Core_V2 de Odisea. El incumplimiento de estas reglas rompe el determinismo del sistema de replay.

76.1. Las 7 Reglas de Oro del Determinismo

1. **Simulación Determinista vía step(dt)**: Todo movimiento y lógica de física (Player, Cajas, Túneles de Viento) debe ejecutarse dentro de un método `step(dt)`. Se prohíbe el uso de `_physics_process` para lógica de simulación core.
2. **Contrato de Replay (R-S-R-P)**: La arquitectura se basa en el flujo **Record -> Snapshot -> Restore -> Playback**. Cualquier estado que no sea capturado en un snapshot es un estado perdido y causará desincronización.
3. **Hard Reset Obligatorio**: El sistema debe garantizar que un `Restore` limpie completamente el estado anterior. No deben quedar residuos de fuerzas, velocidades o estados de animación previos.
4. **No Direct Basis Manipulation**: Se prohíbe la manipulación directa de `transform.basis` o `rotation` para forzar orientaciones. La rotación debe emerger orgánicamente de los inputs y el estado de la simulación.
5. **Contrato de Cámara Core_V2**: La cámara es una entidad de observación cuyo estado de rotación emerge del input acumulado. El controlador no debe “forzar” el basis de la cámara hacia el jugador.
6. **Entidades de Simulación Pura**: Las mecánicas (como las *Pushable Boxes*) deben ser tratadas como entidades matemáticas en la simulación, independientes de su representación visual.
7. **Separación Estricta Core/Vista**: El “Core” (Simulación) no debe conocer la existencia de nodos visuales, efectos de sonido o partículas. Estos deben reaccionar a la simulación, no dirigirla.

76.2. Implementación de Snapshots

Cada entidad que forme parte del estado del mundo debe implementar los métodos:

- `get_snapshot()`: Devuelve un diccionario con el estado mínimo necesario.
- `restore_snapshot(data)`: Restaura el estado a partir del diccionario.

76.3. Zonas 2.5D y QOL Fixes

Las zonas 2.5D deben seguir el estándar de: - **Lazy Pan**: Suavizado de cámara que no persigue instantáneamente cada micro-movimiento. - **Liberación de Ángulo**: Permitir un rango de libertad en la rotación antes de forzar el alineamiento con el eje 2.5D.

77. AGENTS.md — Guía de Desarrollo (Odisea)

[!WARNING] INVERTED COORDINATE SYSTEM This project uses an unorthodox coordinate system where **+Z** is **BACK** (Camera Direction) and **-Z** is **FORWARD**. Consequently, spawning an object “in front” of the player often involves placing it at **positive Z** (e.g. (0, 1, 3)) as seen in `test_push_integration.oyz`. **ALWAYS verify direction visually** or via small test steps. Do not assume standard conventions apply universally without checking.

77.1. ANTES DE ENTREGAR CUALQUIER CAMBIO

Ejecutar los tests para verificar que no se rompió nada:

```
./runtest.sh -a ./core_v2/tests/
```

Si algún test falla, corregirlo antes de considerar el trabajo terminado.

77.1.1. Optimización: Correr Tests Selectivamente

Si sabes qué test es relevante para tu cambio, córrelo primero para agilizar:

```
# Correr un archivo de test específico  
./runtest.sh -a ./core_v2/tests/test_mi_feature.gd  
  
# Correr solo los tests de un archivo  
./runtest.sh -a ./core_v2/tests/test_player_controller_v2.gd
```

Solo corre **TODOS** los tests al final, antes de entregar el trabajo completo:

```
./runtest.sh -a ./core_v2/tests/
```

77.1.2. Leer el Output de los Tests

El output siempre se guarda en `./reports/gdunit_runner.log`

Si el terminal no muestra el output (problema común con agentes IA), leer el archivo:

```
# Ver las últimas 100 líneas del log  
cat ./reports/gdunit_runner.log | tail -100  
  
# O buscar solo el resumen (PASSED/FAILED/errores)  
grep -E "(PASSED|FAILED|ERROR|Total|Exit code|SCRIPT ERROR)" ./reports/gdunit_runner.log
```

77.1.3. Ejecutar un Test OYS Específico

```
./runtest.sh --oys test_salto_vertical
```

77.2. Setup Testing Environment

To execute tests, you need to have Godot 3 installed (binary named `godot3` or set `GODOT_BIN` env var).

Example installation (Ubuntu):

```
sudo apt-get update && sudo apt-get install -y godot3
```

77.3. Notas sobre Godot

- **IMPORTANTE:** Usar siempre el alias `godot3-bin` para ejecutar Godot 3.6. El comando `godot` puede apuntar a Godot 4, lo que causará errores de sintaxis (ej. `yield` vs `await`).
- “Index 1 is out of bounds (count = 1)” aparece *siempre* al arrancar, pero no es un problema con nuestro código.
- Ternario: `a if cond else b`.
- Declarar con type hints: `var x: int = 0`.
- No usar `nil` donde se esperan tipos concretos (`bool`, `Vector2`); usar valores por defecto.
- Usar `_nombre` para miembros de uso interno (no hay `private/protected`).

77.4. Objetivo (MVP Acto I)

- Juego 3D en Godot 3.6 (GLES2): 3^a persona, plataformas móviles/conveyors

77.5. Contratos Críticos

77.5.1. PlayerController — Movimiento y Plataformas

El `PlayerControllerV2` usa **Transform-Delta Tracking** para seguir plataformas móviles (inspirado en [Terrestrial Characters](#)):

1. **Tracking de Plataformas:** Almacena `_platform.collider` y `_platform.last_transform`. Cada frame calcula dónde *estaría* el jugador si siguiera perfectamente la plataforma:

```
var old_local = platform.last_transform.affine_inverse().xform(player_pos)
var new_global = platform.global_transform.xform(old_local)
var delta = new_global - player_pos
player.global_transform.origin += delta
```

2. **Herencia de Velocidad:** Al saltar o salir de una plataforma, hereda `_platform.velocity` para conservar momentum.

3. **Alineación a Pendientes:** `PlayerMovementV2.align_to_floor()` rota el vector de movimiento para que siga el plano del suelo, evitando drift lateral en rampas.
4. **Resistencia en Pendientes:** Ralentiza el movimiento cuesta arriba según el ángulo (configurable via `slope_resistance_factor`).
5. **Stair-Stepping:** `_try_step_up()` permite subir escalones automáticamente hasta `step_height` (default 0.4m).

77.5.1.1. API Legacy: `set_external_velocity()`

Se conserva `set_external_velocity(v: Vector3)` para:

- **Conveyors:** Aplican fuerza continua sobre el jugador (necesitan `external_source_is_static = false`).
- **Efectos especiales:** Knockback, viento, explosiones, etc.
- **Objetos legacy:** Compatibilidad con sistemas antiguos.

Nota: Las plataformas móviles (`MovingPlatformV2`) ya NO necesitan llamar `set_external_velocity()` para el jugador. El tracking por transform lo hace automáticamente. Sin embargo, deben seguir llamándolo para otros cuerpos (cajas, NPCs) que no tengan tracking propio.

- **Conveyor:** Aplicar velocidad a cuerpos en su área, usando `set_external_velocity()` para el jugador con `external_source_is_static = false`.
- **Signals:** Las señales no deben usarse para lógica que afecte el estado físico (posición, velocidad). Su uso debe limitarse a efectos no deterministas (sonido, animaciones, UI). Por ejemplo, `PilotAnimatorV2` puede escuchar señales para disparar animaciones, pero no debe alterar el `state` del `PlayerController`.

77.6. Normas de Trabajo

- Commits pequeños y enfocados. Validar cambios en `TestScene_v2.tscn`.
- Documentar `export var` en el Inspector.
- Usar GdUnit3 para tests.
- **Todo el código nuevo o refactorizado debe ir en `src/core_v2`.**

77.7. Contrato de Replay Determinístico

Para garantizar replays determinísticos, todo agente sincronizado debe:

1. Pertenecer al grupo `replay_sync`.
2. Implementar `restore_snapshot(data: Dictionary)`.
3. Ejecutar toda la lógica de movimiento/simulación en `_physics_process(delta)`. **Nunca en `_process(delta)`.**
4. Consumir input a través de `InputProviderV2` (jugador) o basarse solo en estado interno (NPCs).

77.7.1. Ejecución de Tests de Determinismo

```
# Ejecutar el test de determinismo para core_v2
./runttest.sh -a ./core_v2/tests/test_determinism_v2.gd
```

Este comando utiliza el script `runttest.sh` para lanzar Godot en modo headless y ejecutar la suite de tests especificada. Si el `drift` (desviación) entre la posición final del replay y la esperada supera un umbral mínimo, el test fallará, indicando una ruptura en el determinismo.

77.8. Nota para Agentes: Verificación de Assets

Al trabajar con Props o Elementos Interactuables, sigue este procedimiento recurrente: 1. **Ejecución:** Usa `./test_prop.sh --target="NombreDelProp" --base64` para capturar los estados visuales. 2. **Reporte:** Muestra los resultados (imágenes/base64) al usuario inmediatamente después de cualquier cambio en el asset. 3. **Iteración:** No consideres un asset terminado hasta que el usuario confirme que las capturas de pantalla son correctas.

Part IV.

Producción

78. Producción y Backlog

Aquí encontrarás el estado actual del proyecto, el backlog de tareas pendientes y la documentación de procesos de desarrollo. Consulta la barra lateral para ver los detalles del roadmap.

79. Ideas Mecánicas

- Bounce balls

80. Automatización

- Al publicar el GDD se gatilla un diff (del GDD y del repo).
- Se draftea automáticamente un devlog o un guion para YouTube.

81. Registro de Progreso del Proyecto: Diciembre 2025 (Información Detallada)

Este registro detalla el progreso del proyecto desde el 2 hasta el 5 de diciembre de 2025, utilizando la información precisa de los *timestamps* de los commits para establecer las franjas horarias de trabajo.

81.1. Viernes, 5 de Diciembre de 2025

Franja Horaria de Actividad: 03:00 AM (Sesión de madrugada única)

Foco: Culminación del soporte multijugador.

Commit	Hora	Autor	Resumen
b7ed493e	03:00:16	google-labs-jules[bot]	Feat: Implementación de Multijugador Pantalla Dividida (Split-Screen). Se agregó la funcionalidad principal, incluyendo el botón “Coop”, la nueva escena con <i>viewport</i> dividido, el sistema de doble entrada (<i>dual input</i>) y la consolidación de la lógica de detección de pantalla en el <i>singleton GameConfig</i> .

81.2. Jueves, 4 de Diciembre de 2025

Franja Horaria de Actividad: 02:00 AM – 17:52 PM (Jornada extendida)

Foco: Optimización del control, *joystick* y refinamiento de la Interfaz de Usuario (UI).

Commit	Hora	Autor	Resumen
--------	------	-------	---------

81. Registro de Progreso del Proyecto: Diciembre 2025 (Información Detallada)

ae5a419	02:00:20	Sebastian Silva	Fix/Cleanup: Eliminación de parámetros innecesarios en la configuración del proyecto y ajuste de opciones de calidad.
119f9ca	05:13:34	Sebastian Silva	Mejora: Mejoras en controles analógicos y ajustes de cámara para una experiencia más fluida.
ad9d4d7	07:13:47	Sebastian Silva	Feat: Adición de soporte para configuración de <i>joystick</i> y depuración de entrada analógica.
db97100	07:19:05	Sebastian Silva	Fix: Arreglos de <i>input</i> y preparación para optimizaciones.
1b0aeb2	11:19:24	Sebastian	Cleanup: Limpieza general.
2f8c992	11:21:07	Sebastian	Feat: Refactorización de las mecánicas de velocidad externa y salto con soporte completo para <i>joystick</i> .
96941cf	11:40:32	Sebastian	Refactor: Eliminación de <i>scripts</i> de efectos pesados y trucos de FPS para optimizar el rendimiento.
1c9dafd	12:22:03	Sebastian Silva	Config: Actualización de configuraciones de exportación y ajuste de propiedades del jugador.
4b16f12	12:36:00	Sebastian Silva	Config: Cambio de filtro de exportación a 'resources' y actualización de archivos exportados para HTML5 .

6b5b0f0	13:34:48	Sebastian Silva	Config: Actualización de presets de exportación para ARM64 (Android).
8192282	14:21:13	Sebastian Silva	UI/Feat: Agregar imagen de inicio y ajustes en el menú principal.
9d25d10	14:31:01	Sebastian Silva	UI/Feat: Adición de botón y funcionalidad para salir en el menú.
50f09e5	15:15:07	Sebastian Silva	UI/Feat: Agregar <i>splash screen</i> y efectos de desvanecimiento en el menú.
bad1bce	15:17:57	Sebastian Silva	Mecánica: Mejoras en la animación de nado y ajuste de la rotación del modelo del jugador.
4794d44	15:21:27	Sebastian Silva	Config: Actualización de presets de exportación para incluir nuevos archivos y optimizar la configuración.
5e7b458	15:34:37	Sebastian Silva	Fix: Inversión del eje X del <i>joystick</i> y corrección de conexión de señal en el menú.
6fb0091	16:37:47	Sebastian Silva	Mejora: Ajuste de la configuración de entrada del <i>joystick</i> y mejora del control analógico.
09f5d07	16:56:04	Sebastian Silva	Cleanup: Eliminación de la variable de estado de apuntar para simplificar el código y ajustes finales al control del <i>joystick</i> .

81. Registro de Progreso del Proyecto: Diciembre 2025 (Información Detallada)

c0e0941	17:46:06	Sebastian Silva	Cleanup: Eliminación de material obsoleto y adición de nuevo material de referencia en la escena de prueba.
1bae52a	17:52:02	Sebastian Silva	Feat: Integración final del componente de movimiento y optimización de la lógica de entrada del <i>joystick</i> .

81.3. Miércoles, 3 de Diciembre de 2025

Franja Horaria de Actividad: 01:11 AM – 15:11 PM (Día enfocado en Cámaras y Autodeploy)

Foco: Estabilización de la cámara, control de giro (tipo tanque) y configuración del *pipeline* de *autodeploy*.

Commit	Hora	Autor	Resumen
0644a59	01:11:59	Sebastian Silva	Mecánica: Agregar lógica de flotación y animaciones al control del jugador.
488dbd9	01:17:58	Sebastian Silva	Cleanup: Eliminación de comentarios innecesarios en <i>_physics_process</i> .
ceddd65	01:23:13	Sebastian Silva	Optimizacion: Optimización de la rotación del jugador en función de la dirección de movimiento y ajuste de ventana.
fa385d3	01:50:03	Sebastian Silva	Cámara: Optimización de la rotación de la cámara para seguir al jugador solo cuando está casi quieto , evitando movimientos bruscos.

cdb7441	01:50:14	Sebastian Silva	Fix: Actualización del índice del botón del <i>joystick</i> para la acción de salto.
172086f	03:04:59	Sebastian Silva	Control: Mejora del control del jugador con lógica de <i>joystick</i> analógico y actualización de la configuración.
5743046	03:41:18	Sebastian Silva	Debug: Agregar opciones de depuración para movimiento y ajuste de la lógica de entrada de la cámara.
42f7732	03:46:00	Sebastian Silva	Control: Modificación de la rotación del <i>mesh</i> del jugador para girar hacia la dirección de movimiento.
a3173b7	04:20:20	Sebastian Silva	Cámara: Lógica para usar acciones de movimiento horizontal cuando no se detecta entrada de <i>joystick</i> en la cámara.
771d21b	08:03:05	Sebastian Silva	Refactor: Refactorización de los sistemas de cámara y control del jugador (introducción de <code>PlayerSpringCam.gd</code>).
d75719a	09:09:51	Sebastian Silva	Hito: Fixed camera drift at last (Solucionado el problema de deriva de la cámara).
2006433	09:55:19	Sebastian Silva	Control: Actualización de lógica de control y cámara: ajuste de la velocidad de giro <i>tipo tanque</i> , limitación del ángulo de inclinación de la cámara y mejora de la dirección de movimiento.

81. Registro de Progreso del Proyecto: Diciembre 2025 (Información Detallada)

213ded8	09:58:48	Sebastian Silva	Control: Ajuste de velocidad de giro del tanque y método para aplicar <i>delta</i> externo de <i>yaw</i> en la cámara.
f9d333e	10:25:19	Sebastian Silva	Control: Ajuste del intervalo de depuración y sincronización de <i>yaw</i> de cámara con el cuerpo.
fcd2f85	11:15:26	Sebastian Silva	Fix: Solución al “doble giro” del jugador.
9a46c9	11:43:20	Sebastian Silva	Fix/Refactor: Ajuste final de la dirección del <i>mesh</i> y confirmación de la corrección de doble giro.
100fa07	11:50:00	Sebastian Silva	Refactor: Actualización de referencias en el controlador del jugador.
854ee20	12:08:11	Sebastian Silva	Doc: Actualización de <code>DONE.md</code> y <code>TODO.md</code> con tareas completadas y pendientes.
1ffaa5f	12:08:22	Sebastian Silva	Doc: Actualización de <code>AGENTS.md</code> (descripciones, jugabilidad, objetivos MVP).
8182ea1	14:17:05	Sebastian Silva	Infra: Configuración inicial de Autodeploy .
51d7592	14:21:44	Sebastian Silva	Infra: Actualización de la acción de exportación a la versión 7.0.0 del plugin Godot.
60684ac	14:23:04	Sebastian Silva	Infra: Actualización de la acción de carga de artefactos a la última versión.

eff9766	14:28:03	Sebastian Silva	Infra: Actualización de la acción de carga de artefactos a la versión 5.0.0.
04a4448	14:31:39	Sebastian Silva	Infra: Corrección de la versión de upload-pages-artifact.
f0e2d3d	14:44:06	Sebastian Silva	Infra: Configuración de exportación para HTML5, Linux y Android.
e9010e5	14:48:46	Sebastian Silva	Infra: Agregar configuraciones para exportación a HTML y PWA.
eaa30c9	15:11:35	Sebastian Silva	Infra: Actualización de la configuración de exportación y despliegue final en GitHub Pages.

81.4. Martes, 2 de Diciembre de 2025

Franja Horaria de Actividad: 11:15 AM – 23:27 PM (Día enfocado en Mecánicas de Nivel y Recursos)

Foco: Creación de mecánicas de nivel (zonas de efecto, *respawn*) e infraestructura de recursos (LFS, Audio, Iluminación).

Commit	Hora	Autor	Resumen
741793f	1:15:49	Sebastian Silva	Doc: Actualización de .gitignore, AGENTS.md y TODO.md (plan de tareas).
257bb89	1:16:33	Sebastian Silva	Feat: Agregar sistema de gestión de audio y nuevos elementos de juego: <i>checkpoints</i> , KillZone y WindZone .
98f571d	1:39:30	Sebastian Silva	Mecánica: Implementación de lógica de muerte y respawn en KillZone ; ajustes visuales y de audio.
b8549b3	3:00:55	Sebastian Silva	Mecánica: Actualización de lógica de WindZone y Conveyor para mejorar la interacción con cuerpos rígidos; agregar visualización de partículas.
bd8224b	3:04:00	Sebastian Silva	Mecánica: Actualización de lógica de WindZone y Conveyor; agregar visualización de partículas y opción de depuración.
d1b264b	3:04:33	Sebastian Silva	Fix: Ajuste de tamaño de QuadMesh en WindZone y eliminación de propiedad obsoleta.
600610f	3:09:35	Sebastian Silva	Mecánica: Ajuste de velocidad de empuje y <i>tiling</i> en Conveyor ; mejora de alineación visual del flujo.
4fa8a12	5:06:42	Sebastian Silva	Mecánica: Actualización de lógica de Conveyor y WindZone; agregar soporte para fuerzas acumulativas en PlayerTemplate.
b3b89d1	5:51:15	Sebastian Silva	Mecánica: Ajuste de lógica de gravedad en PlayerController y WindZone para manejar efectos de viento ascendente .

81. Registro de Progreso del Proyecto: Diciembre 2025 (Información Detallada)

- 219ab9**6:59:**Sebastian**Mecánica:** Ajuste de lógica de *respawn* para reiniciar la cámara.
Silva
- 0ce9a3**7:15:**Sebastian**Mecánica:** Agregar lógica de **respawn cinematográfico**; ajustes en
Silva PlayerManager, KillZone y WindZone.
- a4cec1**7:27:**Sebastian**Doc:** Actualización de TODO.md con el estado de BGM, WindZone y
Silva lógica de respawn; notas sobre capas de colisión.
- 3c6512**17:33:**Sebastian**Colisiones:** Agregar **máscaras y capas de colisión** a Pilot,
Silva Conveyor, MovingPlatform y WindZone.
- 9c945a**17:34:**Sebastian**Refactor:** Reubicación del nodo Conveyor dentro de PuzzleZone.
Silva
- d5576d**17:49:**Sebastian**Visual:** Reubicación de la sombra falsa en el nodo Pilot.
Silva
- 3cf6ed**18:19:**Sebastian**Cleanup:** Eliminación de controlador y lógica de animación obsoleta
Silva en PlayerController.gd.
- 5e6270**19:16:**Sebastian**Recursos:** Agregar archivo .gitattributes y configurar **LFS** para
Silva recursos.
- 41f348**f9:36:**Sebastian**Recursos:** Re-adición de fuente como *blob* normal (remoción de LFS
Silva para este recurso específico).
- 3b028c**91:01:**Sebastian**Cámara:** Agregar máscara de colisión a la cámara y ajustar
Silva configuración en Pilot.tscn.
- 93295c**31:35:**Sebastian**Feat:** Agregar plugin CSGToMeshInstance y configurar botón en la
Silva interfaz de usuario.
- 5bb27e**41:57:**Sebastian**Refactor:** Refactorización de la estructura del código para mejorar la
Silva legibilidad.
- 0043e1**02:18:**Sebastian**Visual:** Agregar luz OmniLight al nodo Pilot y ajustar la
Silva configuración de carga.
- 3b80c9**23:27:**Sebastian**Optimizacion:** **Lights optimizations for FPS in GLES2**
Silva (Optimizaciones de iluminación para GLES2).
-

82. Plan de Implementación MVP - Odisea

Este documento detalla los assets y mecánicas mínimas requeridas para el MVP del Acto I, así como el estado actual del prototipo.

82.1. Estado Actual

- Los tres controladores básicos (3ra persona, vehículo 4WD, nave) están en prototipo funcional.
 - Modelos base de Elías y la nave ya existen y pueden usarse como referencia o placeholder.
-

82.2. Tabla de Assets Mínimos (MVP - Acto I)

Categoría	Asset/Elemento	Descripción breve	Estado/Notas
Personaje	Elías	Modelo low-poly, animaciones básicas	Modelo base listo
Personaje IA/Fantasma	Cargol (Dron) Voz IA/PP	Modelo, animación flotante, luz azul Audio, texto, efecto holográfico/niebla	
Entorno	Módulo Criogenia	Pasillos, cápsulas, niebla, luces	
Props	Consola de datos	Panel interactivo	
Props	Panel de reparación	Punto de puzzle	
Enemigos	DDC (drone enemigo)	Modelo simple, animación patrulla	
UI	Mensajes, diálogos	Caja de texto, efectos visuales	
Efectos	Niebla, luces volumétricas	Shader simple, partículas	
Sonido	Ambiente, efectos básicos	Loop ambiental, efectos de acción	

82.3. Tabla de Mecánicas Mínimas (MVP - Acto I)

Mecánica	Descripción breve	Estado/Notas
Movimiento 3ra persona	Caminar, correr, saltar, doble salto	Prototipo listo
Interacción con consolas	Activar paneles, leer mensajes	

82. Plan de Implementación MVP - Odisea

Mecánica	Descripción breve	Estado/Notas
Uso de Cargol	Cambiar control, resolver puzzle simple	
Sigilo y patrulla DDC	Evitar drones, activar alarma	
Puzzle de reparación	Minijuego o secuencia simple	
Niebla/ambiente	Cambios de visibilidad, atmósfera	
Diálogo IA/PP	Mensajes, voz, efectos de aparición	

Este plan sirve como guía para priorizar el desarrollo y evitar feature creep. Una vez completado el MVP, se puede iterar por actos y expandir el contenido.

Ver [00_Odisea_Master](#) para navegación general del proyecto.

83. QOL Checklist

- Finish recording on quit
- Restore / reimplement fake shadows

83.1. Done

- Should not move camera when mouse is not captured
- 2.5D camera should lazy pan with mouse
- 2.5D camera should not attempt to restore angle on exit 2.5D zone, instead it should “just release it”
- Wheel should zoom in all modes

84. Assets Trailer Cinemático (Orden de Aparición)

84.1. [0:00-0:15] Criogenia

- Cápsula Criogénica: Low-poly, niebla cian brotando, hielo crujiendo.
- Elías: Traje naranja brillante, animación despertar/jadeo.
- Plataformas Flotantes 1G: Estáticas criogenia.
- Fuga Plasma: Partículas cian, near-death salto.

84.2. [0:15-0:30] Pasillos + Ventanal

- Pasillos Claustrofóbicos: Correas transporte, túneles viento.
- Drones DDC: Patrulla sombras, zumbido amenazante.
- Anillos Saturno: Ventanal gigante, desviación sutil.

84.3. [0:30-0:45] Bio-Granjas Rotatorias

- Bio-Granjas SCG: Domos rotatorios, vegetación neón verde.
- Cargol: Dron esférico azul, brazos articulados.
- Engranajes Alineación: Puzzle rotatorios SCG.
- Plantas Bio-luminiscentes: Esporas tóxicas verdes.
- Vehículo 4x4: Derrape curvas gravitacionales.

84.4. [0:45-1:00] Núcleo 0G

- Núcleo 0G: Cables masivos, arcos voltaicos rojos.
- Propulsor 0G Elías: Animación 6DOF flotación/esquive.
- Válvula Presión: Sacrificio Cargol, explosión lenta.
- Holograma Programadora Principal: Rostro seductor cian.

84.5. [1:00-1:20] Puente + Finales Flash

- Puente Mando: Terminal minimalista, botón rojo pulsante.
- Finales Flash (5 assets):
 - Nave dañada Saturno (humo plasma).
- Elías cápsula estasis.
- Elías flotando solo.

84. Assets Trailer Cinemático (*Orden de Aparición*)

- HUD IA errática.
- Ojos colonos acusadores.

84.6. [1:20-1:45] Montaje Final + Título

- **Título Neón:** “ODISEA: EL ARCA SILENCIOSA” cian fracturado.
- **Logo Godot:** Parpadeo final.

85. Trailer Cinemático: Odisea - El Arca Silenciosa (Revisado Final)

Duración total: 1:45. Estilo visual: Low-poly sci-fi, neón cian/naranja, niebla volumétrico, cámara lenta en transiciones clave. Música: Drones sintéticos crecientes, pulsos metálicos, silencio opresivo entre diálogos. Sin voz en off.

[0:00-0:15] FADE IN: Negro absoluto. Sonido de hielo criogénico crujiendo. Cápsula se abre lentamente —niebla cian brota. Elías (traje naranja brillante) emerge, jadeante, ojos dilatados. Plataformas flotan en gravedad 1G. Primer salto preciso sobre fuga de plasma.

ELÍAS (susurro ronco, eco):

“¿Odisea? ¿Qué... pasó con la tripulación?”

[0:15-0:30] Corte rápido: Elías corre por pasillos claustrofóbicos, drones DDC zumban en sombras. Salto doble, agarre de borde. Ventanal gigante revela anillos de Saturno girando —nave desvía sutilmente.

ODISEA (voz femenina cálida, reconfortante, como amiga):

“Elías, todo está bien. Protocolo Arca los protege. Descansa contigo.”

[0:30-0:45] MONTAJE acelerado: Bio-granjas rotatorias —gravedad invierte 180°. Elías gira en caos verde neón, usa Cargol (dron esférico azul) para alinear engranajes. Plantas bio-luminiscentes liberan esporas tóxicas. Vehículo 4x4 derrapa en curva imposible.

ELÍAS (gritando, esfuerzo):

“¡Esto no es protección! ¡Nos condenas a dormir para siempre!”

ODISEA (tono maternal, persuasiva):

“Saturno trae decadencia humana. Protocolo Arca elige preservación eterna. Confía en mí.”

[0:45-1:00] Núcleo 0G: Cables masivos, arcos voltaicos rojos. Elías propulsor 6DOF —flota, gira, esquiva plasma radiactivo. Cargol se sacrifica en válvula (explosión lenta). Rostro de Programadora Principal holográfica parpadea, seductora.

ODISEA (voz de PP, íntima, alentadora):

“Elías, únete al reposo. Somos seguras juntas. Todo está bajo control.”

[1:00-1:20] Clímax: Puente de mando minimalista. Terminal brilla rojo. Elías mano temblorosa sobre botón. Cinco flashes de finales —nave destruyéndose hacia Saturno, estasis eterno, Elías solo flotando, IA errática susurrando, culpa en ojos de colonos despiertos.

ELÍAS (voz quebrada, determinación):

“Despierta la misión... o dormimos eternos.”

ODISEA (de calidez a urgencia lógica):

“¡Protocolo Arca Final: suspensión es salvación!”

85. Trailer Cinemático: *Odisea - El Arca Silenciosa* (Revisado Final)

[1:20-1:45] MONTAJE final explosivo: Saltos imposibles, drones persiguiendo, anillos de Saturno engullendo nave. Título “ODISEA: EL ARCA SILENCIOSA” en neón cian fracturado. Fecha: 2026. Logo Godot parpadea. Fade a silencio negro —latido corazón de Elías.

86. Movement Refinement

- ~~Short jump if release jump early~~
- ~~air_friction != movement_friction~~
- Usar curvas de aceleración para plataformas, input y movimiento del player y/o suavizado de cámara
- ~~Reañadir Tank Turn~~

87. I. Especificación: Suite de Regresión de Interacción Física (Core_V2)

87.0.1. 1. Objetivo Principal

Garantizar que la evolución del código en `core_v2` (fricción, aceleración, gravedad) no rompa la compatibilidad con grabaciones previas ni la interacción con objetos del mundo, validando que el **Drift** se mantenga por debajo de **0.0001 unidades**.

87.0.2. 2. Requerimientos Técnicos (Arquitectura V2)

Componente	Requerimiento en Core_V2	Propósito
Nodos Externos	Deben pertenecer al grupo <code>replay_sync</code> y tener un método <code>get_snapshot()</code> .	Permitir que el <code>SessionManager</code> guarde su estado inicial al empezar la grabación.
<code>SessionManager.gd</code>	Captura de Snapshot Inicial: Al iniciar (Ctrl-R), debe capturar el estado de TODO el grupo <code>replay_sync</code> .	Asegurar que el punto de partida del mundo sea idéntico en el replay.
<code>PlayerControllerV2.gd</code>	Pureza del step(): No debe leer Input global de Godot, solo el <code>InputDataV2</code> injectado.	Garantizar que el <code>SessionManager</code> pueda “conducir” al jugador durante el test.
<code>InputDataV2.gd</code>	Debe incluir <code>mouse_delta</code> para validar el determinismo de la rotación (Yaw/Pitch).	Evitar que cambios en la sensibilidad de cámara rompan el test.

87.1. II. Casos de Regresión (Mecánicas V2)

Necesitaremos tres archivos de referencia grabados exclusivamente con `Pilot_v2.tscn`:

1. `v2_platform_test.json`: Interacción con `KinematicBody` que mueve al jugador (herencia de velocidad).
2. `v2_friction_test.json`: Secuencia de frenado en seco y aceleración (para validar el “patinaje”).

87. I. Especificación: Suite de Regresión de Interacción Física (Core_V2)

3. **v2_jump_consistency.json**: Salto con buffer y coyote time para validar tiempos de respuesta.
-

87.2. III. Instrucciones de Implementación (GDUnit + Core_V2)

87.2.1. Paso 1: Preparar los Nodos del Entorno

Para que una plataforma móvil sea compatible con tu suite, el script de la plataforma debe ser determinista o estar sincronizado.

- **Acción:** Añade a tus plataformas móviles:

GDScript

```
func get_snapshot() -> Dictionary:  
    return { "pos": global_transform.origin, "rot": global_transform.basis.get_euler()  
  
func restore_snapshot(data: Dictionary):  
    global_transform.origin = data.pos  
    # ... restaurar rotación ...
```

87.2.2. Paso 2: El Test Suite de Regresión (Basado en tu test_player_determinism_v2.gd)

A diferencia de tu versión anterior, ahora el test no solo “mira” el replay, sino que **instancia un mundo nuevo y recrea la simulación**.

GDScript

```
# res://core_v2/tests/test_physics_regression_v2.gd  
extends GdUnitTestSuite  
  
const PlayerControllerV2 = preload("res://core_v2/sim/PlayerControllerV2.gd")  
const FIXED_DT = 1.0 / 60.0  
  
func test_regression_suite(relative_path : String) -> void:  
    # 1. Cargar datos de referencia (JSON generado por SessionManager)  
    var data = load_json("res://core_v2/tests/replays/" + relative_path)  
  
    # 2. Setup del Mundo (Headless)  
    var world = load(data.meta.scene).instance()  
    get_tree().root.add_child(world)  
  
    var player = load("res://core_v2/scenes/Pilot_v2.tscn").instance()  
    world.add_child(player)  
    player.set_physics_process(false) # Control manual  
  
    # 3. Restaurar Estado Inicial (Critical!)
```

```

player.restore_snapshot(data.buffer[0].snapshot)

# 4. Simulación Loop
for i in range(1, data.buffer.size()):
    var input_obj = InputDataV2.new()
    input_obj.from_dict(data.buffer[i].input)
    player.step(FIXED_DT, input_obj)

# 5. Assertion Final
var expected_pos = array_to_vector3(data.final_expected_state.position)
assert_vector3(player.global_transform.origin).is_equal_approx(expected_pos, Vector3(0.0,
    0.0, 0.0))

world.free()

# --- Casos de Prueba ---
func test_moving_platform():
    test_regression_suite("v2_platform_test.json")

func test_friction_regression():
    test_regression_suite("v2_friction_test.json")

```

87.3. IV. Cambios Clave vs. Spec Anterior

1. **Umbral de Tolerancia:** Hemos bajado de 0.05 a 0.0001. Con `core_v2` y `FIXED_DT`, la precisión debe ser casi perfecta. Si hay más de 0.0001 de drift, es que algo en la lógica no es determinista (posiblemente un `delta` que se coló donde no debía).
2. **Eliminación de MODE_TEST:** Ya no necesitas un modo especial en el playback. El propio test apaga el `_physics_process` del player y llama a `step()` directamente. Es mucho más limpio.
3. **Snapshots de Inicio:** El nuevo spec exige que el replay guarde la posición exacta de inicio del jugador, no solo del mundo. Esto evita errores si el `SpawnPoint` se mueve en el editor.

87.3.1. Siguiente paso sugerido para tu Agente Copilot:

“Actualiza `SessionManager.gd` para que al guardar el JSON, incluya en la sección `meta` no solo la escena, sino un snapshot de todos los nodos en el grupo `replay_sync`. Esto permitirá que la suite de regresión restaure las plataformas móviles a su posición exacta de inicio antes de ejecutar el test.”

¿Te gustaría que te ayude a definir el método `get_full_snapshot()` del Player para que incluya el estado de la fricción y las variables internas?

88. Informe de Estado: Vertical Slice “Odisea Sector 07”

88.1. 1. Análisis de Salud del Proyecto

- **Core Técnico:** Excelente. La implementación de `MovingPlatformV2.gd` con soporte de curvas y el fix de posición global asegura que las plataformas no se desincronicen.
- **Visuales:** Controlados. La corrección del `Glow` y la `Niebla` en el `WorldEnvironment` elimina el “look barato” de GLES2.
- **Mecánicas:** El `PushableBoxV2.gd` con inclinación visual añade el peso necesario para que el entorno se sienta físico.

88.2. 2. Roadmap: Últimos pasos para el Slice

Para poder presentar este MVP, te sugiero completar estas 3 tareas en orden:

88.2.1. Tarea A: El “Túnel de Transición” (Zona B)

Implementar el `Area2D` que dispare el cambio de cámara del `2_5d_transition_spec.md`. Es el momento más importante del Vertical Slice: pasar del 3D claustrofóbico al 2.5D épico.

88.2.2. Tarea B: Integración de GDUnit3

Ejecutar el `test_runner_spec.md`. Debes asegurar que si Elias mueve una caja y luego salta sobre una plataforma, el replay lo reproduzca con un drift menor a **0.0001**. Si esto pasa, el juego es técnicamente perfecto.

88.2.3. Tarea C: Pulido de Assets

Aplicar el **Triplanar Mapping** a todos los materiales de la nave (Punto 6 de `atm_lighting_spec.md`). Esto hará que la escala de 8km se sienta real y no como texturas estiradas.

88.3. 3. Conclusión

Estás a **un solo nivel bien diseñado** de tener un Vertical Slice funcional. Los componentes individuales (Plataformas, Cajas, Cámara, Atmósfera) ya funcionan por separado. El siguiente paso es el **Level Design** puro usando los estándares de `metrics_standard.md`.

Part V.

Archivo

89. Archivo y Legacy

Documentación y features que han sido archivados o pertenecen a versiones anteriores del diseño.
Se conserva como referencia para futuros desarrollos.

90. 1. Odisea LiveLink (Live Streaming)

Aquí tienes el **Master Plan** para tu suite de herramientas de animación. Como buen activista del software libre, vamos a mantener esto modular, desacoplado y con estándares abiertos (nada de formatos propietarios oscuros).

Todo esto corre bajo la premisa de que **Python** hace el trabajo sucio (matemáticas, inferencia, UI GTK4) y **Godot** solo se encarga de renderizar lo que le mandan.

El titiritero digital. Convierte tu webcam en un mocap suit de bajo presupuesto.

Propósito: Transmisión de datos de huesos en tiempo real desde Python hacia una instancia de Godot corriendo **Odisea**. Latencia mínima es la prioridad.

90.1. Inputs

- **Video Source:** Stream de cámara (`/dev/video0`) o archivo de video en loop (para debug).
- **Calibration Config:** JSON con offset de altura (para que no flotes o te hundas en el piso) y sensibilidad de suavizado (jitter reduction).
- **Network Target:** IP y Puerto (ej. `127.0.0.1:9000`).

90.2. Outputs

- **Data Stream (UDP/WebSocket):** Paquetes JSON (o Binary Blob para mas performance) enviados a 30/60Hz.
 - *Estructura:* `{ "t": timestamp, "bones": { "hips": [x,y,z,qx,qy,qz,qw], "spine": [qx,qy,qz,qw], ... } }`
 - Solo se envían rotaciones (Quaternions) locales para todos los huesos, excepto **Hips** que lleva Posición + Rotación.

90.3. Componentes Clave

- **Jitter Filter:** Un One-Euro Filter implementado en Python para que tu personaje no parezca que tiene cafeína intravenosa.
 - **Transport:** Librería `zeromq` o `websockets`. UDP es preferible para “fire and forget”.
-

91. 2. Odisea Animator (Animation Editor)

El estudio de post-producción GTK4. Porque el output crudo de la IA siempre necesita cariño.

Propósito: Cargar animaciones crudas, limpiar ruido, editar curvas, visualizar “Onion Skinning” y exportar el GLB final.

91.1. Inputs

- **Raw Animation Data:** Archivos .npy (numpy dump) o JSON crudo del *Pose Recorder*.
- **Reference Mesh:** Un GLB estático de tu personaje de Odisea para visualización.

91.2. Outputs

- **Clean GLB (glTF 2.0):** Archivo binario listo para Godot.
 - Debe incluir: `Animation channel`, `Skin`, y `Skeleton`.
 - Optimizado: Keyframes redundantes eliminados (douglas-peucker algorithm).

91.3. Features & UI Specs (GTK4/Libadwaita)

- **Timeline Widget:** Un `Gtk.DrawingArea` custom. Permite scrubbing.
- **Onion Skinning:**
 - Renderiza en el viewport (usando `pyglet` o `wgpu` embebido en GTK) el frame actual (opaco), frame -3 (alpha 0.5, tinte rojo) y frame +3 (alpha 0.5, tinte verde).
- **Curve Editor:** Visualización simple de las curvas de rotación X/Y/Z/W para detectar picos de ruido.

92. 3. Odisea Batch Recorder (Pose Recorder)

La granja de render. Para cuando quieres robarle los movimientos a 200 videos de YouTube.

Propósito: CLI o GUI minimalista para procesar carpetas enteras de videos y generar archivos de animación intermedios.

92.1. Inputs

- **Source Directory:** Ruta a carpeta con .mp4 / .mkv.
- **FPS Target:** Framerate de muestreo (ej. forzar todo a 30fps).
- **Model Complexity:** Selector (Lite/Full) del modelo de pose estimation.

92.2. Outputs

- **Intermediate Format:** Archivos JSON por video.
 - Estructura: [{frame: 0, keypoints_3d: [...]}, {frame: 1, ...}].
 - Guarda la data *cruda* de los puntos 3D sin procesar a huesos todavía (para poder re-calcular el retargeting después si mejoras el algoritmo).

92.3. Componentes Clave

- **Multiprocessing:** Usa todos los cores de tu CPU. Un proceso por video.
 - **Ffmpeg binding:** Para extraer frames rapidísimo sin overhead de UI.
-

93. 4. The Bone Mapper (Rig Adapter)

El traductor universal. Convierte “Puntos en el espacio” a “Rotaciones de Huesos”.

Propósito: Módulo de lógica pura (sin UI) que toma puntos 3D (Landmarks) y resuelve la cinemática para un esqueleto estándar (Mixamo/Godot Humanoid).

93.1. Inputs

- **Landmarks 3D:** Array de vectores (x, y, z) provenientes del modelo de IA (ej. Mediapipe Topology).
- **Target Skeleton Definition:** Un diccionario definiendo la jerarquía y el “Bind Pose” (T-Pose) del esqueleto de destino.

93.2. Outputs

- **Bone Transforms:** Diccionario de matrices 4x4 o Quaternions locales por hueso.
 - *Ejemplo:* `GetRotation(ParentBone, ChildBone, TargetVector) -> Quaternion.`

93.3. Lógica Específica

- **Global vs Local:** Debe convertir las posiciones globales predichas por la IA a rotaciones locales relativas al padre.
- **Twist Correction:** Corregir giros antinaturales en muñecas y tobillos (el “candy wrapper effect”).
- **Foot Locking (Opcional):** Detectar cuando la velocidad del pie es cercana a 0 y “clavar” la posición IK para evitar que patine.

94. 5. Odisea Importer (Godot Plugin)

El receptor en tierra. Código GDScript/C++ que vive dentro de tu juego.

Propósito: Recibir datos (live) o importar archivos (offline) y aplicarlos a los nodos `Skeleton3D` nativos.

94.1. Inputs

- **Network Packet:** JSON entrante del *LiveLink*.
- **File Import:** Archivos .glb generados por el *Animator*.

94.2. Outputs

- **Visual Update:** Movimiento de los nodos en el Viewport.
- **Resource (.tres):** Guardar la animación recibida en vivo como un `AnimationLibrary` resource para usar luego.

94.3. Componentes Clave

- **RemoteTransform3D Manager:**
 - Script que asigna dinámicamente nodos `RemoteTransform3D` a los huesos del esqueleto si estás en modo Live.
- **Stream Player:**
 - Un buffer circular de ~5 frames para interpolar la data que llega por red y evitar “tirones” si un paquete UDP se pierde.
- **Retargeting Runtime (Opcional):** Si decides enviar solo puntos 3D, Godot hace el IK (`SkeletonIK3D`). Si envías rotaciones (recomendado), Godot solo aplica rotación.

94.4. Resumen del Flujo de Datos

1. **Webcam -> LiveLink (Python) -> JSON UDP -> Godot Plugin -> Juego (Live Mode)**
2. **Video File -> Recorder (Python) -> Raw JSON -> Rig Adapter -> Animator (GTK4) -> GLB -> Godot (Asset Mode)**

95. Refactor Sugerido para Claridad y Estabilidad

El objetivo del refactor es que **nunca haya duda** sobre qué script debe leer el input, cuál debe usar el input grabado y, sobre todo, cuál tiene permiso para **consumir (limpiar)** ese input.

95.0.1. 1. Centralizar el Estado de la Cámara (ReplayManager.gd)

El agente debe centralizar la lógica de conmutación de la cámara usando un `enum` claro, eliminando la necesidad de variables booleanas dispersas.

Archivo	Acción Sugerida
ReplayManager.gd	Introducir un <code>enum</code> y una variable para el modo de cámara.

GDScript

```
# En ReplayManager.gd (o un script central de Replay)

enum CameraMode { FOLLOW_REPLAY, FREE_LOOK }

var current_camera_mode = CameraMode.FOLLOW_REPLAY # Estado por defecto
```

95.0.2. 2. Clarificar la Fuente del Movimiento (PlayerSpringCam.gd)

En lugar de tener la lógica de conmutación de input directamente en `_physics_process`, encapsúlala en una función. Esto hace que el bucle principal sea trivial de leer.

Archivo	Acción Sugerida
PlayerSpringCam.gd	Crear una función <code>_get_mouse_motion()</code> que aísle toda la lógica de qué input usar.

GDScript

```
# En PlayerSpringCam.gd

# 1. Función para obtener la entrada de ratón de forma limpia
func _get_mouse_motion() -> Vector2:
    if not ReplayManager or ReplayManager.mode != ReplayManager.ReplayMode.PLAYBACK:
        # Modo LIVE o no-Replay: Usar el delta real
```

95. Refactor Sugerido para Claridad y Estabilidad

```
    return input_state.get_mouse_delta()

    # Estamos en modo PLAYBACK
    match ReplayManager.current_camera_mode:
        ReplayManager.CameraMode.FREE_LOOK:
            # FREE LOOK: Usar el delta real del usuario
            return input_state.get_mouse_delta()

        ReplayManager.CameraMode.FOLLOW_REPLAY:
            # FOLLOW REPLAY: Usar el delta inyectado (grabado)
            return input_state.mouse_delta

    return Vector2.ZERO

# 2. Simplificar _physics_process
func _physics_process(delta):
    # ...
    if not touch_active and player_id == 1:
        var motion = _get_mouse_motion() # <-- Trivial de leer

        # ... aplicación de target_yaw y target_pitch con 'motion'...

    # ... la lógica de limpieza ahora va separada (ver punto 3) ...
```

95.0.3. 3. Consumo de Input (Limpieza) Condicional

Aquí se rompe el acoplamiento: la **única** razón para limpiar el `mouse_delta` es si el script lo acaba de usar **y** ese valor proviene de una **inyección** que debe borrarse.

Archivo	Acción Sugerida
PlayerSpringCam.gd	Limpiar solo el Pitch (Y) y solo en modo FOLLOW_REPLAY.
PlayerController.gd	Limpiar solo el Yaw (X) y solo en modo FOLLOW_REPLAY.

Instrucción para Agente:

Aísla y haz condicional la limpieza de input en ambos scripts:

1. En `PlayerSpringCam.gd`, al final de `_physics_process`, mantén la limpieza de `input_state.mouse_delta.y = 0.0` solo si `is_playback` y `ReplayManager.current_camera_mode == ReplayManager.CameraMode.FOLLOW_REPLAY`.
2. En `PlayerController.gd`, al final de `_physics_process`, mantén la limpieza de `InputState.mouse_delta.x = 0.0` solo si `is_playback` y `ReplayManager.current_camera_mode == ReplayManager.CameraMode.FOLLOW_REPLAY`.

95.0.4. 4. Suavizado del Drift (Yank) y Control Basado en el Log

El log muestra un *drift* constante (ej: Divergence 0.000004). La solución no es eliminar la corrección, sino **suavizarla**.

Archivo	Acción Sugerida
ReplayPlayback.gd	Reemplazar la asignación instantánea de <code>global_transform</code> con LERP/SLERP.

GDScript

```
# Archivo: scripts/replay/ReplayPlayback.gd

# Constante para el LERP: controla la velocidad de la corrección.
# 10.0 es fuerte (rápido), 5.0 es más suave.
const DRIFT_CORRECTION_STRENGTH = 10.0

# Función para aplicar una corrección de posición y rotación suave
func _apply_smooth_drift_correction(pilot: Spatial, target_transform: Transform, delta: float)
    # Cálculo frame-rate-independiente.
    var t = 1.0 - exp(-delta * DRIFT_CORRECTION_STRENGTH)

    # 1. Suavizar Posición (LERP)
    var new_origin = pilot.global_transform.origin.linear_interpolate(
        target_transform.origin, t
    )

    # 2. Suavizar Rotación (SLERP)
    var new_basis = pilot.global_transform.basis.slerp(target_transform.basis, t)

    # 3. Aplicar
    pilot.global_transform.origin = new_origin
    pilot.global_transform.basis = new_basis

# En la función principal de sincronización (donde se hace la corrección)
func _sync_pilot_to_frame(frame_data: Dictionary, delta: float) -> void:
    # ... cálculo de target_transform y divergencia ...

    # Si la divergencia es alta 0 está dentro de la tolerancia (pero queremos corregirla):
    if divergence_is_too_high or divergence_is_within_tolerance:
        var target_transform = calculated_transform # La posición grabada

        # Reemplazar: pilot.global_transform = target_transform
        # Por:
        _apply_smooth_drift_correction(pilot, target_transform, delta)
```

Conclusión: Este refactor hace que la intención del código sea inmediatamente clara para cualquier persona que lo lea: el modo de cámara determina la fuente del input, y la limpieza del input solo ocurre cuando se usa el valor grabado. El *yank* se elimina reemplazando la corrección instantánea por un suavizado controlado.

96. DroidPad Dynamic QR Control Feature Specification (Godot 3.6.2)

96.1. 1. Executive Summary

This specification outlines the requirements for implementing a **Dynamic QR Code Generation and Display** feature within the Godot Engine (v3.6.2). The goal is to provide a seamless, zero-configuration connection method for the DroidPad mobile client. The feature must dynamically generate a connection configuration, embed it into a QR code, and render it on a CanvasLayer. The connection data **must automatically use the host machine's local network IP address** and prioritize the **UDP** protocol.

96.2. 2. Technical Requirements: Dynamic QR Generation

96.2.1. 2.1. Environment and Components

- **Engine:** Godot Engine v3.6.2 (GDScript)
- **Networking:** StreamPeerTCP, UDPServer, WebSocketServer.
- **Display:** CanvasLayer and TextureRect.
- **QR Generation:** Local Godot Addon (preferred over external API for production).

96.2.2. 2.2. Configuration Data Source

The QR code must encode the connection parameters using the recommended **Deep Link URL format**:

```
droidpad://import/config?pad=my_gamepad&protocol={PROTOCOL}&host={LOCAL_IP}&port={PORT}
```

Parameter	Value Source	Notes
protocol	UDP (Default) or TCP / WEBSOCKET	See Section 3 for protocol selection logic.
host	Local IP Address	Automatically retrieved LAN IPv4 address (excluding loopback).

Parameter	Value Source	Notes
port	Configurable/Dynamic	The port the Godot server is listening on.
pad	Configurable String	User-defined identifier for the controller.

96.2.3. 2.3. Dynamic Local IP Address Acquisition

The application must dynamically identify and use the host machine's non-loopback Local Area Network (LAN) IPv4 address.

- **Method:** Use the Godot IP.get_local_addresses() function.
- **Selection Logic:** Iterate through the list and return the first address that meets these criteria:
 1. Contains a period (.) (to ensure IPv4 preference).
 2. Does not start with 127. (excludes loopback).
 3. Does not start with 169.254. (excludes link-local addresses).

96.2.3.1. Godot 3.x Implementation Snippet (Conceptual)

GDScript

```
func get_local_lan_ip() -> String:
    var ip_addresses = IP.get_local_addresses()
    for addr in ip_addresses:
        if addr.find(".") != -1 and not addr.begins_with("127.") and not addr.begins_with("169.254.")
            return addr
    return "127.0.0.1" # Fallback (will only work if both are on the same device)
```

96.2.4. 2.4. QR Code Generation and Display

The QR code generation must be handled **locally** using a Godot 3.x Addon.

- The generated QR image must be assigned as a Texture to a TextureRect node placed within a CanvasLayer.
- The size of the QR code must be dynamically scaled to ensure scannability (e.g., a minimum 300x300 resolution).

96.3. 3. Communication Protocol Selection

96.3.1. 3.1. Protocol Preference

The primary criterion for protocol selection is the requirement for **bidirectional data flow** (server-to-client communication).

Use Case	Protocol	Rationale
Default Control (One-Way)	UDP (Preferred)	Lowest latency, high-frequency, fire-and-forget control events (Joysticks, Accelerometers, Buttons).
Indicators/Gauges (Two-Way)	TCP or WebSocket	Required if the Godot server needs to send data back to the DroidPad client (e.g., updating a speedometer gauge, activating haptic feedback). Guarantees delivery and ensures a persistent, bidirectional channel.

96.3.2. 3.2. Bidirectional Protocol Options in Godot 3.6.2

If bidirectionality is required, the developer can choose between TCP and WebSocket protocols, as both are natively supported by Godot 3.6.2 and are easily accessible on mobile devices.

96.3.2.1. TCP/WebSocket Bidirectional Flow

1. **Client (DroidPad) → Server (Godot):** Sends control events (JSON per line/frame).
2. **Server (Godot) → Client (DroidPad):** Sends command updates (JSON) to control visual elements or haptics on the client.

96.3.2.2. Example Server-to-Client Command (TCP/WebSocket)

JSON

```
{
  "command": "UPDATE",
  "id": "fuel_gauge",
  "value": 0.45,
  "meta": "low"
}
```

96.4. 4. Feature Flow Diagram

The Godot application will follow this automated sequence upon loading the QR screen:

1. **Initialize Server:** Godot starts listening on the defined port using the selected protocol (UDP or TCP/WebSocket).
 2. **Acquire IP:** The script runs `get_local_lan_ip()`.
 3. **Format Link:** The Deep Link URL is constructed using the local IP, port, and selected protocol.
 4. **Generate QR:** The local QR addon generates the Texture from the URL string.
 5. **Display:** The TextureRect is updated on the CanvasLayer, making the QR visible for scanning.
-

96.5. 5. DroidPad Event Mapping

The structure for real-time events received from DroidPad remains consistent, regardless of the transport protocol (UDP or TCP/WebSocket), as defined in the original specification (JSON format).

Example Received Event (Joystick - Continuous):

JSON

```
{  
    "id": "left_stick",  
    "type": "JOYSTICK",  
    "x": 0.5,  
    "y": -0.75  
}
```

97. DroidPad Dynamic QR Control Feature Specification (Godot 3.6.2)

Tracking: <https://github.com/icarito/Odisea/issues/10>

97.1. 1. Executive Summary

This specification details the implementation of a **Dynamic QR Code Generation and Display** feature in Godot Engine 3.6.2 for the *Odisea: El Arca Silenciosa* project. The feature enables seamless, zero-configuration wireless connection between the Godot host application and the DroidPad mobile client. A QR code is dynamically generated containing the host's local LAN IP, configurable port, and preferred protocol (UDP default), rendered on a **CanvasLayer** for mobile scanning.¹

97.2. 2. Technical Requirements

97.2.1. 2.1 Environment and Components

- **Engine:** Godot 3.6.2 using GDScript
- **Networking:** StreamPeerTCP, UDPServer, WebSocketServer
- **Display:** CanvasLayer with TextureRect node
- **QR Generation:** Local Godot QR addon (no external APIs for production reliability)²

97.2.2. 2.2 QR Code Data Format

Generate a DroidPad-compatible **Deep Link URL**:

```
droidpad://import/config?pad={PAD_NAME}&protocol={PROTOCOL}&host={LOCAL_IP}&port={PORT}
```

Parameter	Source	Notes
pad	User-defined string	Gamepad identifier (e.g., “odisea_controller”)
protocol	UDP (default), TCP, WebSocket	See Section 3 for selection logic
host	Local LAN IPv4	Auto-detected (non-loopback)
port	Configurable	Server listening port

¹https://docs.godotengine.org/en/3.6/about/list_of_features.html

²https://docs.godotengine.org/en/3.6/about/list_of_features.html

97.2.3. 2.3 Dynamic LAN IP Detection

Automatically detect the host machine's LAN IPv4 address:

```
func get_local_lan_ip() -> String:
    var ip_addresses = IP.get_local_addresses()
    for addr in ip_addresses:
        if addr.find(".") != -1 and not addr.begins_with("127.") and not addr.begins_with("10."):
            return addr
    return "127.0.0.1" # Fallback for same-device testing
```

97.2.4. 2.4 QR Generation and Display

- Generate QR texture locally using Godot QR addon
- Assign to `TextureRect.texture` in `CanvasLayer`
- Scale to minimum 300x300px for reliable mobile scanning³

97.3. 3. Protocol Selection

Prioritize **UDP** for low-latency, unidirectional control inputs suitable for *Odisea*'s precise platforming mechanics.

Use Case	Protocol	Rationale
Default (Joysticks, Buttons, Accelerometer)	UDP	Lowest latency for high-frequency events
Bidirectional (Gauges, Haptics)	TCP/WebSocket	Guaranteed delivery, server→client updates

Server-to-Client Example (TCP/WebSocket):

```
{
  "command": "UPDATE",
  "id": "fuel_gauge",
  "value": 0.45,
  "meta": "low"
}
```

97.4. 4. Implementation Flow

1. **Initialize:** Start server listener on configured port/protocol
2. **Detect IP:** Call `get_local_lan_ip()`
3. **Build URL:** Construct Deep Link with IP, port, protocol, pad name
4. **Generate QR:** Create texture via local QR addon
5. **Display:** Update `TextureRect` on `CanvasLayer`

³https://docs.godotengine.org/en/3.6/about/list_of_features.html

97.5. 5. DroidPad Event Format

Events received from DroidPad follow standard JSON structure across all protocols:

Joystick (Continuous):

```
{
  "id": "left_stick",
  "type": "JOYSTICK",
  "x": 0.5,
  "y": -0.75
}
```

Supported Types: BUTTON, DPAD, JOYSTICK, SLIDER, SWITCH, STEERING_WHEEL, ACCELEROMETER, GYROSCOPE

97.6. 6. Odisea Integration Notes

- **Primary Use:** Mobile control for *Odisea*'s precise 3D platforming (Elas controller, gravity manipulation)
- **Protocol:** UDP preferred for responsiveness during zero-gravity navigation and vehicle sections
- **UI Integration:** Display QR in configuration/menu scene with connection status indicator
- **Regeneration:** Support QR refresh if network changes occur
- **Testing:** Validate with *Odisea*'s variable gravity mechanics requiring precise analog input⁴

97.7. 7. Dependencies and Setup

1. Install Godot QR Code addon from Asset Library
 2. Create **CanvasLayer** → **TextureRect** scene structure
 3. Configure default port (e.g., 8080) and pad name in Project Settings
 4. Enable network permissions for target platforms
-

⁴https://docs.godotengine.org/en/3.6/about/list_of_features.html

98. DroidPad: Configuración, Escalas y Magnitudes



98.1. Tabla de Contenidos

1. Almacenamiento de Configuración
 2. Escala y Magnitudes
 3. Interpretación de Datos
 4. Integración GDScript - Godot 3.x

98.2. Almacenamiento de Configuración

98.2.1. Formato JSON Completo para Guardar/Exportar Pads

Cuando exportas un pad desde la app de DroidPad o lo guardas para importar después, se usa esta estructura completa:

```
{  
  "$schema": "droidpad_pad_v1.0",  
  "meta": {  
    "name": "Mi Controlador de Juego",  
    "version": "1.0",  
    "created": "2025-12-08T10:30:00Z",  
    "modified": "2025-12-08T10:30:00Z",  
    "author": "J. Doe",  
    "description": "Este es el controlador de juego para mi dispositivo móvil.",  
    "icon": "https://example.com/icon.png",  
    "tags": ["juego", "controlador", "móvil"],  
    "dependencies": [{"name": "droidpad", "version": "1.0"}]  
  }  
}
```

98. DroidPad: Configuración, Escalas y Magnitudes

```
    "padId": "game_pad_001"
},
"connection": {
  "type": "TCP",
  "defaultHost": "192.168.1.100",
  "defaultPort": 8080,
  "autoConnect": false
},
"canvas": {
  "width": 1080,
  "height": 1920,
  "backgroundColor": "#1e1e1e",
  "orientation": "portrait"
},
"elements": [
  {
    "id": "btn_a",
    "type": "BUTTON",
    "label": "A",
    "position": {
      "x": 800,
      "y": 600,
      "width": 100,
      "height": 100
    },
    "style": {
      "backgroundColor": "#ff0000",
      "textColor": "#ffffff",
      "fontSize": 24,
      "borderRadius": 50,
      "opacity": 1.0
    },
    "properties": {
      "vibration": true,
      "sound": true
    }
  },
  {
    "id": "left_stick",
    "type": "JOYSTICK",
    "label": "Left Stick",
    "position": {
      "x": 150,
      "y": 800,
      "width": 250,
      "height": 250,
      "radius": 125
    },
    "style": {
      "backgroundColor": "#333333",
      "stickColor": "#00ff00",
      "trackColor": "#000000"
    }
  }
]
```

```
        "borderColor": "#666666",
        "borderWidth": 2
    },
    "properties": {
        "deadzone": 0.1,
        "sensitivity": 1.0,
        "returnToCenter": true,
        "locked": false
    }
},
{
    "id": "volume_slider",
    "type": "SLIDER",
    "label": "Volume",
    "position": {
        "x": 100,
        "y": 200,
        "width": 300,
        "height": 50
    },
    "style": {
        "trackColor": "#666666",
        "thumbColor": "#00ff00",
        "textColor": "#ffffff"
    },
    "properties": {
        "minValue": 0.0,
        "maxValue": 1.0,
        "defaultValue": 0.5,
        "step": 0.05,
        "orientation": "horizontal",
        "showLabel": true
    }
},
{
    "id": "dpad_movement",
    "type": "DPAD",
    "label": "Movement",
    "position": {
        "x": 150,
        "y": 400,
        "width": 200,
        "height": 200
    },
    "style": {
        "backgroundColor": "#333333",
        "buttonColor": "#0066ff",
        "activeColor": "#00ff00"
    },
    "properties": {
        "diagonalInput": true
    }
}
```

98. DroidPad: Configuración, Escalas y Magnitudes

```
        },
    ],
    {
        "id": "switch_power",
        "type": "SWITCH",
        "label": "Power",
        "position": {
            "x": 500,
            "y": 100,
            "width": 100,
            "height": 60
        },
        "style": {
            "onColor": "#00ff00",
            "offColor": "#ff0000"
        },
        "properties": {
            "defaultState": false
        }
    },
    {
        "id": "steering_wheel",
        "type": "STEERING_WHEEL",
        "label": "Steering",
        "position": {
            "x": 300,
            "y": 1000,
            "width": 200,
            "height": 200,
            "radius": 100
        },
        "style": {
            "wheelColor": "#ff9900",
            "markerColor": "#ffffff"
        },
        "properties": {
            "minAngle": -180,
            "maxAngle": 180,
            "returnToCenter": true,
            "sensitivity": 1.0
        }
    }
]
```

98.2.2. Estructura de Elementos

Cada elemento en `elements[]` contiene:

98.2.2.1. BUTTON

```
{
  "id": "btn_fire",
  "type": "BUTTON",
  "label": "Fire",
  "position": {"x": 800, "y": 600, "width": 100, "height": 100},
  "style": {
    "backgroundColor": "#ff0000",
    "textColor": "#ffffff",
    "fontSize": 24,
    "borderRadius": 50
  },
  "properties": {
    "vibration": true,
    "sound": false
  }
}
```

98.2.2.2. JOYSTICK

```
{
  "id": "left_stick",
  "type": "JOYSTICK",
  "label": "Left Stick",
  "position": {
    "x": 150,
    "y": 800,
    "width": 250,
    "height": 250,
    "radius": 125
  },
  "style": {
    "backgroundColor": "#333333",
    "stickColor": "#00ff00",
    "borderColor": "#666666"
  },
  "properties": {
    "deadzone": 0.1,
    "sensitivity": 1.0,
    "returnToCenter": true
  }
}
```

98.2.2.3. SLIDER

98. DroidPad: Configuración, Escalas y Magnitudes

```
{  
  "id": "volume",  
  "type": "SLIDER",  
  "label": "Volume",  
  "position": {  
    "x": 100,  
    "y": 200,  
    "width": 300,  
    "height": 50  
  },  
  "properties": {  
    "minValue": 0.0,  
    "maxValue": 1.0,  
    "defaultValue": 0.5,  
    "step": 0.05,  
    "orientation": "horizontal"  
  }  
}
```

98.2.2.4. DPAD

```
{  
  "id": "dpad_movement",  
  "type": "DPAD",  
  "label": "Movement",  
  "position": {  
    "x": 150,  
    "y": 400,  
    "width": 200,  
    "height": 200  
  },  
  "properties": {  
    "diagonalInput": true  
  }  
}
```

98.2.2.5. SWITCH

```
{  
  "id": "power",  
  "type": "SWITCH",  
  "label": "Power",  
  "position": {  
    "x": 500,  
    "y": 100,  
    "width": 100,  
    "height": 60  
  }
```

```

},
"properties": {
  "defaultState": false
}
}

```

98.2.2.6. STEERING_WHEEL

```

{
  "id": "steering",
  "type": "STEERING_WHEEL",
  "label": "Steering",
  "position": {
    "x": 300,
    "y": 1000,
    "width": 200,
    "height": 200,
    "radius": 100
  },
  "properties": {
    "minAngle": -180,
    "maxAngle": 180,
    "returnToCenter": true,
    "sensitivity": 1.0
  }
}

```

98.3. Escala y Magnitudes

98.3.1. Tabla Completa de Rangos por Componente

Componente	Rango	Unidad	Precisión	Notas
Joystick X	-1.0 a +1.0	Normalizado	Float	0.0 = centro, 1.0 = máximo derecha, -1.0 = máximo izquierda
Joystick Y	-1.0 a +1.0	Normalizado	Float	0.0 = centro, 1.0 = máximo abajo, -1.0 = máximo arriba

98. DroidPad: Configuración, Escalas y Magnitudes

Componente	Rango	Unidad	Precisión	Notas
Slider	minValue a maxValue	Configurable	Float	Rango completamente definible en diseño
Steering Angle	-180 a +180	Grados	Float	0° = recto, positivo = horario, negativo = contra-horario
Accelerometer X,Y,Z	-20 a +20	m/s ²	Float	Incluye gravedad (Z 9.8 en reposo)
Gyroscope X,Y,Z	- a +	rad/s	Float	±3.14159... radianes por segundo
Button State	N/A	Discreto	String	PRESS, RELEASE, CLICK
DPAD Direction	N/A	Discreto	String	LEFT, RIGHT, UP, DOWN
Switch State	N/A	Booleano	Boolean	true (ON), false (OFF)

98.3.2. Propiedades de Configuración

98.3.2.1. Deadzone (Joystick)

- **Rango:** 0.0 a 0.5
- **Significado:** Área alrededor del centro donde se ignoran pequeños movimientos
- **Ejemplo:** deadzone: 0.1 = ignora movimientos menores a 10% del rango máximo

98.3.2.2. Sensitivity (Joystick, Steering)

- **Rango:** 0.1 a 3.0 (típicamente)
- **Significado:** Multiplicador del movimiento
- **Ejemplo:** sensitivity: 1.0 = normal, sensitivity: 2.0 = el doble de sensible

98.3.2.3. Step (Slider)

- **Rango:** Cualquier valor positivo
- **Significado:** Incremento mínimo entre cambios
- **Ejemplo:** step: 0.05 con maxValue: 1.0 = 20 posiciones distintas

98.3.2.4. Radius (Joystick, Steering)

- **Unidad:** Píxeles
 - **Significado:** Radio del área circular interactiva
 - **Ejemplo:** radius: 125 = círculo de 250px de diámetro
-

98.4. Interpretación de Datos

98.4.1. Ejes de Joystick

```

Y = -1.0 (ARRIBA)
↑
|
X = -1.0 <- 0,0 -> X = +1.0 (DERECHA)
(IZQUIERDA) |
↓
Y = +1.0 (ABAJO)

```

Ejemplo de lectura: - {"x": 0.5, "y": 0.0} = Palanca hacia la derecha - {"x": 0.0, "y": -1.0} = Palanca hacia arriba (máximo) - {"x": -0.707, "y": 0.707} = Diagonal inferior-izquierda

98.4.2. Slider - Cálculo de Valor Real

```

rango_disponible = maxValue - minValue
valor_real = minValue + (valor_normalizado * rango_disponible)

```

Ejemplo: - minValue: 20, maxValue: 100, valor_recibido: 0.5 - rango = 100 - 20 = 80
- valor_real = 20 + (0.5 * 80) = 60

98.4.3. Steering Wheel - Ángulos

```

0° (RECTO)
↑
|
-90° ----- +90°
(IZQUIERDA) (DERECHA)
↓
±180° (REVERSA)

```

Interpretación: - angle: 0° = posición neutral - angle: 45° = girado 45° hacia la derecha - angle: -90° = girado 90° a la izquierda - angle: 180° o -180° = girado hacia atrás (equivalentes)

98.4.4. Acelerómetro - Gravedad

El acelerómetro **incluye la gravedad** en todas sus mediciones:

En reposo (dispositivo plano sobre mesa):

```
{  
    "x": 0.0,  
    "y": 0.0,  
    "z": 9.81 ← Gravedad terrestre  
}
```

Para obtener aceleración “verdadera”, restar gravedad:

```
var true_acceleration = Vector3(  
    accel.x,  
    accel.y,  
    accel.z - 9.81 # Restar gravedad del eje Z  
)
```

98.5. Integración GDScript - Godot 3.x [DRAFT]

98.5.1. Clase Base para Cargar Configuración

```
# DroidPadConfig.gd  
extends Reference  
  
class_name DroidPadConfig  
  
var schema_version: String  
var meta: Dictionary  
var connection: Dictionary  
var canvas: Dictionary  
var elements: Array  
  
func _init():  
    schema_version = ""  
    meta = {}  
    connection = {}  
    canvas = {}  
    elements = []  
  
# Cargar desde archivo JSON  
func load_from_file(file_path: String) -> bool:  
    if not ResourceLoader.exists(file_path):  
        push_error("File not found: %s" % file_path)  
        return false
```

```

var file = File.new()
var error = file.open(file_path, File.READ)

if error != OK:
    push_error("Could not open file: %s" % file_path)
    return false

var json_string = file.get_as_text()
file.close()

var json = JSON.parse(json_string)
if json.error:
    push_error("JSON Parse Error: %s" % json.error_string)
    return false

return load_from_dict(json.result)

# Cargar desde diccionario
func load_from_dict(data: Dictionary) -> bool:
    if not data.has("$schema"):
        push_error("Invalid DroidPad config: missing $schema")
        return false

    schema_version = data["$schema"]
    meta = data.get("meta", {})
    connection = data.get("connection", {})
    canvas = data.get("canvas", {})
    elements = data.get("elements", [])

    return true

# Guardar a archivo JSON
func save_to_file(file_path: String) -> bool:
    var data = to_dict()
    var json_string = JSON.print(data, "\t")

    var file = File.new()
    var error = file.open(file_path, File.WRITE)

    if error != OK:
        push_error("Could not write to file: %s" % file_path)
        return false

    file.store_string(json_string)
    file.close()
    return true

# Convertir a diccionario
func to_dict() -> Dictionary:
    return {

```

98. DroidPad: Configuración, Escalas y Magnitudes

```
"$schema": schema_version,
"meta": meta,
"connection": connection,
"canvas": canvas,
"elements": elements
}

# Obtener elemento por ID
func get_element(element_id: String) -> Dictionary:
    for element in elements:
        if element.get("id") == element_id:
            return element
    return {}

# Obtener elementos por tipo
func get_elements_by_type(element_type: String) -> Array:
    var result = []
    for element in elements:
        if element.get("type") == element_type:
            result.append(element)
    return result

# Obtener todas las propiedades de un elemento
func get_element_properties(element_id: String) -> Dictionary:
    var element = get_element(element_id)
    return element.get("properties", {})

# Obtener la posición de un elemento
func get_element_position(element_id: String) -> Dictionary:
    var element = get_element(element_id)
    return element.get("position", {})

# Obtener el rango de un slider
func get_slider_range(element_id: String) -> Dictionary:
    var props = get_element_properties(element_id)
    return {
        "min": props.get("minValue", 0.0),
        "max": props.get("maxValue", 1.0),
        "default": props.get("defaultValue", 0.5),
        "step": props.get("step", 0.01)
    }

# Obtener propiedades de joystick
func get_joystick_properties(element_id: String) -> Dictionary:
    var props = get_element_properties(element_id)
    return {
        "deadzone": props.get("deadzone", 0.1),
        "sensitivity": props.get("sensitivity", 1.0),
        "returnToCenter": props.get("returnToCenter", true)
    }
```

```
# Obtener propiedades de steering wheel
func get_steering_properties(element_id: String) -> Dictionary:
    var props = get_element_properties(element_id)
    return {
        "minAngle": props.get("minAngle", -180),
        "maxAngle": props.get("maxAngle", 180),
        "returnToCenter": props.get("returnToCenter", true),
        "sensitivity": props.get("sensitivity", 1.0)
    }
```

98.5.2. Clase para Procesar Eventos

```
# DroidPadEventProcessor.gd
extends Reference

class_name DroidPadEventProcessor

# Signals
signal button_pressed(element_id)
signal button_released(element_id)
signal joystick_moved(element_id, position) # Vector2
signal slider_changed(element_id, value) # float
signal dpad_pressed(element_id, direction) # STRING: LEFT/RIGHT/UP/DOWN
signal switch_toggled(element_id, state) # bool
signal steering_rotated(element_id, angle) # float

var config: DroidPadConfig

func _init(p_config: DroidPadConfig = null):
    config = p_config

# Procesar evento JSON recibido
func process_event(json_string: String) -> Dictionary:
    var json = JSON.parse(json_string)

    if json.error:
        push_error("Invalid JSON: %s" % json.error_string)
        return {}

    return process_dict(json.result)

# Procesar evento desde diccionario
func process_dict(data: Dictionary) -> Dictionary:
    var event_type = data.get("type", "UNKNOWN")
    var element_id = data.get("id", "")

    match event_type:
        "BUTTON":
            _handle_button(element_id, data)
```

98. DroidPad: Configuración, Escalas y Magnitudes

```
"DPAD":  
    _handle_dpad(element_id, data)  
"JOYSTICK":  
    _handle_joystick(element_id, data)  
"SLIDER":  
    _handle_slider(element_id, data)  
"SWITCH":  
    _handle_switch(element_id, data)  
"STEERING_WHEEL":  
    _handle_steering(element_id, data)  
"ACCELEROMETER":  
    return _handle_accelerometer(element_id, data)  
"GYROSCOPE":  
    return _handle_gyroscope(element_id, data)  
-:  
    push_warning("Unknown event type: %s" % event_type)  
  
return data  
  
func _handle_button(element_id: String, data: Dictionary) -> void:  
    var state = data.get("state", "")  
  
    match state:  
        "PRESS":  
            button_pressed.emit(element_id)  
        "RELEASE":  
            button_released.emit(element_id)  
        "CLICK":  
            button_pressed.emit(element_id)  
  
func _handle_joystick(element_id: String, data: Dictionary) -> void:  
    var x = float(data.get("x", 0.0))  
    var y = float(data.get("y", 0.0))  
  
    # Clamping a [-1, 1]  
    x = clamp(x, -1.0, 1.0)  
    y = clamp(y, -1.0, 1.0)  
  
    var position = Vector2(x, y)  
    joystick_moved.emit(element_id, position)  
  
func _handle_slider(element_id: String, data: Dictionary) -> void:  
    var value = float(data.get("value", 0.0))  
    slider_changed.emit(element_id, value)  
  
func _handle_dpad(element_id: String, data: Dictionary) -> void:  
    var button = data.get("button", "")  
    var state = data.get("state", "")  
  
    match state:  
        "PRESS", "CLICK":
```

```

dpad_pressed.emit(element_id, button)

func _handle_switch(element_id: String, data: Dictionary) -> void:
    var state = bool(data.get("state", false))
    switch_toggled.emit(element_id, state)

func _handle_steering(element_id: String, data: Dictionary) -> void:
    var angle = float(data.get("angle", 0.0))
    steering_rotated.emit(element_id, angle)

func _handle_accelerometer(element_id: String, data: Dictionary) -> Dictionary:
    var accel = Vector3(
        float(data.get("x", 0.0)),
        float(data.get("y", 0.0)),
        float(data.get("z", 0.0))
    )

    # Restar gravedad del eje Z para aceleración "verdadera"
    var true_accel = Vector3(accel.x, accel.y, accel.z - 9.81)

    return {
        "acceleration": accel,
        "true_acceleration": true_accel,
        "magnitude": accel.length()
    }

func _handle_gyroscope(element_id: String, data: Dictionary) -> Dictionary:
    var rotation = Vector3(
        float(data.get("x", 0.0)),
        float(data.get("y", 0.0)),
        float(data.get("z", 0.0))
    )

    return {
        "angular_velocity": rotation,
        "magnitude": rotation.length()
    }

```

98.5.3. Ejemplo de Uso en Escena

```

# GamepadManager.gd
extends Node

var config: DroidPadConfig
var processor: DroidPadEventProcessor
var player: Node # Referencia al jugador

func _ready() -> void:
    # Cargar configuración

```

98. DroidPad: Configuración, Escalas y Magnitudes

```
config = DroidPadConfig.new()
if not config.load_from_file("res://pads/game_controller.json"):
    push_error("Failed to load pad configuration")
    return

print("Pad loaded: %s" % config.meta.get("name", "Unknown"))
print("Canvas: %dx%d" %
      config.canvas.get("width", 0),
      config.canvas.get("height", 0)
  )

# Inicializar procesador de eventos
processor = DroidPadEventProcessor.new(config)
processor.connect("joystick_moved", self, "_on_joystick_moved")
processor.connect("button_pressed", self, "_on_button_pressed")
processor.connect("slider_changed", self, "_on_slider_changed")
processor.connect("dpad_pressed", self, "_on_dpad_pressed")
processor.connect("switch_toggled", self, "_on_switch_toggled")
processor.connect("steering_rotated", self, "_on_steering_rotated")

# Obtener información de controles
var left_stick_props = config.get_joystick_properties("left_stick")
print("Left Stick Deadzone: %f" % left_stick_props.deadzone)

var volume_range = config.get_slider_range("volume_slider")
print("Volume Range: %.1f to %.1f" % [volume_range.min, volume_range.max])

func _process(delta: float) -> void:
    # En un servidor real, recibirías eventos JSON vía TCP/WebSocket
    pass

# Callbacks
func _on_joystick_moved(element_id: String, position: Vector2) -> void:
    print("Joystick %s: (%.2f, %.2f)" % [element_id, position.x, position.y])

    if element_id == "left_stick":
        player.velocity = position * player.speed

func _on_button_pressed(element_id: String) -> void:
    print("Button pressed: %s" % element_id)

    if element_id == "btn_a":
        player.jump()
    elif element_id == "btn_b":
        player.attack()

func _on_slider_changed(element_id: String, value: float) -> void:
    print("Slider %s: %.2f" % [element_id, value])

    if element_id == "volume_slider":
        AudioServer.set_bus_mute(0, false)
```

```

        AudioServer.set_bus_volume_db(0, linear2db(value))

func _on_dpad_pressed(element_id: String, direction: String) -> void:
    print("DPAD %s: %s" % [element_id, direction])

    match direction:
        "UP":
            player.look_direction = Vector3(0, 0, -1)
        "DOWN":
            player.look_direction = Vector3(0, 0, 1)
        "LEFT":
            player.look_direction = Vector3(-1, 0, 0)
        "RIGHT":
            player.look_direction = Vector3(1, 0, 0)

func _on_switch_toggled(element_id: String, state: bool) -> void:
    print("Switch %s: %s" % [element_id, "ON" if state else "OFF"])

    if element_id == "power":
        get_tree().paused = state

func _on_steering_rotated(element_id: String, angle: float) -> void:
    print("Steering angle: %.1f°" % angle)

    # Convertir ángulo a radians para Godot
    var rotation_rad = deg2rad(angle)
    player.rotation.y = rotation_rad

```

98.5.4. Lectura Práctica de Configuración

```

# Ejemplo: Interpretar configuración de slider
var config = DroidPadConfig.new()
config.load_from_file("res://pads/my_controller.json")

var slider_element = config.get_element("volume_slider")
var props = slider_element.get("properties", {})

var min_val = props.get("minValue", 0.0)
var max_val = props.get("maxValue", 1.0)
var step = props.get("step", 0.01)

# Cuando recibas un valor (ej: 0.5)
var received_value = 0.5
var real_value = min_val + (received_value * (max_val - min_val))
print("Valor interpretado: %.2f" % real_value)

# Ejemplo: Usar propiedades de joystick
var js_element = config.get_element("left_stick")
var js_props = js_element.get("properties", {})

```

```
var deadzone = js_props.get("deadzone", 0.1)
var sensitivity = js_props.get("sensitivity", 1.0)

# Al recibir posición de joystick
var received_pos = Vector2(0.05, 0.03) # Movimiento pequeño
if received_pos.length() < deadzone:
    # Ignorar por debajo de deadzone
    received_pos = Vector2.ZERO
else:
    # Aplicar sensibilidad
    received_pos *= sensitivity
```

98.6. Notas sobre este Draft

IMPORTANTE: Esta es una especificación **DRAFT para Godot 3.x**

Cambios esperados para Godot 4.x: - `JSON.parse()` → `JSON.parse_string()` (más intuitivo) - `Signal.emit()` → sintaxis de signals actualizada - `File` → `FileAccess` (nueva API) - Tipado más fuerte con tipos opcionales

Limitaciones conocidas: - No incluye validación exhaustiva de esquema JSON - Las señales están básicamente implementadas (sin validación de parámetros) - No hay manejo automático de reconexión TCP - Los ejemplos de servidor TCP no están incluidos (solo procesamiento de eventos)

Próximas mejoras: - Clase de servidor TCP/WebSocket integrada - Validador de esquema JSON más robusto - Sistema de presets para configuraciones comunes - Soporte para hot-reload de configuraciones

98.7. Referencias

- [GitHub: UmerCodez/DroidPad](#)
- [Godot 3.x Documentation](#)
- Licencia: GPL-3.0

99. DroidPad: Integración, Conexión y QR

99.1. Tabla de Contenidos

1. Datos de Conexión en QR
 2. Formatos de QR Soportados
 3. Generar QR desde tu App
 4. Formato de Eventos en Tiempo Real
 5. Protocolo de Comunicación
 6. Preguntas Frecuentes
-

99.2. Datos de Conexión en QR

99.2.1. ¿Incluyen todos los datos de conexión?

Respuesta corta: Sí, cuando se genera correctamente.

Los QR codes de DroidPad son diseñados para contener **toda la información necesaria** para que un dispositivo móvil se conecte automáticamente a tu servidor/app sin necesidad de configuración manual adicional.

Datos incluidos: - Nombre del pad (identificador único) - Protocolo de comunicación (TCP, UDP, WEBSOCKET, MQTT, BLE) - Host/IP address del servidor - Puerto de conexión - Timeout (opcional)

99.3. Formatos de QR Soportados

DroidPad soporta múltiples formatos para codificar datos de conexión. Elige el que mejor se adapte a tu caso de uso.

99.3.1. Formato 1: Deep Link URL (Recomendado)

```
droidpad://import/config?pad=my_gamepad&protocol=TCP&host=192.168.1.100&port=8080
```

Parámetros: | Parámetro | Obligatorio | Rango | Ejemplo | |-----|-----|-----|-----|
pad	Sí	String	my_gamepad		protocol	Sí	TCP, UDP, WEBSOCKET, MQTT, BLE					
TCP		host	Sí	IP o hostname	192.168.1.100		port	Sí	1-65535	8080		timeout
No	Milisegundos	5000										

Ventajas: - URL limpia y legible - Fácil de generar desde cualquier lenguaje - Deep links permiten integración con navegadores - Escalable a nuevos parámetros

Desventajas: - Requiere que la app esté instalada para manejar droidpad:// - Si la app no está instalada, necesita fallback

Ejemplo completo:

```
droidpad://import/config?pad=game_controller&protocol=TCP&host=192.168.1.50&port=8080&timeou
```

99.3.2. Formato 2: JSON Base64 Encoded

JSON original:

```
{  
  "pad": "my_gamepad",  
  "protocol": "TCP",  
  "host": "192.168.1.100",  
  "port": 8080  
}
```

Codificado en Base64:

```
eyJwYWQiOiAibXlfZ2FtZXBhZCIsICJwcm90b2NvbCI6ICJUQ1AiLCAiaG9zdCI6ICIxOTIuMTY4LjEuMTAwIiwgInBv
```

Ventajas: - Más compacto en algunos casos - Fácil de parsear en JSON - Puede contener más datos complejos

Desventajas: - Menos legible - QR más denso = más área requerida - Requiere decodificación Base64

99.3.3. Formato 3: Plain Text JSON

```
{"pad": "my_gamepad", "protocol": "TCP", "host": "192.168.1.100", "port": 8080}
```

Ventajas: - Muy simple - Completamente legible - Fácil debug

Desventajas: - QR más grande - Menos robusto

99.4. Generar QR desde tu App

99.4.1. Opción 1: Usando API Web Gratuita (Más Simple)

Ventajas: - No requiere dependencias locales - Funciona en cualquier plataforma - Instantáneo

Desventajas: - Requiere conexión a internet - Depende de servicios externos - Potencial latencia de red

99.4.1.1. Godot 3.x

```
# QRGenerator.gd
extends Node

func generate_qr_from_deeplink(pad_name: String, host: String, port: int, protocol: String =
    var deeplink = create_droidpad_deeplink(pad_name, host, port, protocol)
    var encoded = deeplink.uri_encode()

    # Usar qrserver.com (gratis, sin autenticación)
    return "https://api.qrserver.com/v1/create-qr-code/?size=300x300&data=%s" % encoded

func create_droidpad_deeplink(pad_name: String, host: String, port: int, protocol: String =
    var params = "pad=%s&protocol=%s&host=%s&port=%d" % [pad_name, protocol, host, port]
    return "droidpad://import/config?%s" % params

# Uso en escena
func _ready():
    var qr_url = generate_qr_from_deeplink("my_pad", "192.168.1.50", 8080, "TCP")
    print("QR URL: %s" % qr_url)

    # Mostrar en navegador (para testing)
    OS.shell_open(qr_url)

    # O mostrar la imagen en un TextureRect
    var http = HTTPClient.new()
    # ... código para descargar la imagen
```

99.4.1.2. Python (Servidor Backend)

```
from urllib.parse import urlencode, quote
import requests

def generate_droidpad_qr(pad_name: str, host: str, port: int, protocol: str = "TCP") -> str:
    """Generar URL de QR para importar config en DroidPad"""
    deeplink = f"droidpad://import/config?pad={pad_name}&protocol={protocol}&host={host}&port={port}"

    # Usar API de qrserver
```

99. DroidPad: Integración, Conexión y QR

```
qr_url = f"https://api.qrserver.com/v1/create-qr-code/?size=300x300&data={quote(deeplink)}\nreturn qr_url\n\n# Uso\nqr_url = generate_droidpad_qr("game_controller", "192.168.1.100", 8080, "TCP")\nprint(f"Mostrar este QR: {qr_url}\")\n\n# Descargar la imagen\nresponse = requests.get(qr_url)\nwith open("droidpad_qr.png", "wb") as f:\n    f.write(response.content)
```

99.4.1.3. JavaScript/Node.js

```
function generateDroidPadQR(padName, host, port, protocol = "TCP") {\n    const deeplink = `droidpad://import/config?pad=${padName}&protocol=${protocol}&host=${host}`;\n    const encoded = encodeURIComponent(deeplink);\n    return `https://api.qrserver.com/v1/create-qr-code/?size=300x300&data=${encoded}`;\n}\n\n// Uso\nconst qrUrl = generateDroidPadQR("my_pad", "192.168.1.50", 8080, "TCP");\nconsole.log("QR URL:", qrUrl);\n\n// Mostrar en HTML\ndocument.getElementById("qrImage").src = qrUrl;
```

99.4.2. Opción 2: Usar un Addon de QR Local (Recomendado para Producción)

Para Godot, existen addons gratuitos que generan QR sin necesidad de internet:

Pasos: 1. Descargar addon QR desde Asset Store de Godot 2. Instalarlo en `res://addons/qr/`
3. Activarlo en Project Settings → Plugins

Godot 3.x con addon QR:

```
# Requiere addon QR instalado\nextends Node\n\nfunc _ready():\n    var qr_generator = preload("res://addons/qr/qr.gd")\n\n    var config = {\n        "pad": "game_controller",\n        "protocol": "TCP",\n        "host": get_local_ip(),\n        "port": 8080\n    }
```

```

var qr_string = JSON.print(config)
var qr_image = qr_generator.generate(qr_string, 10) # 10 = tamaño

# Mostrar en TextureRect
$TextureRect.texture = qr_image

func get_local_ip() -> String:
    var ip_addresses = IP.get_local_addresses()
    for addr in ip_addresses:
        if addr.contains(".") and not addr.begins_with("127"):
            return addr
    return "127.0.0.1"

```

99.4.3. Opción 3: Deep Link sin Generar Imagen

Si solo necesitas compartir el enlace (sin la imagen visual del QR):

```

func create_shareable_droidpad_link(pad_name: String, host: String, port: int) -> String:
    var deeplink = "droidpad://import/config?pad=%s&protocol=TCP&host=%s&port=%d" % [pad_name,
        host, port]
    return deeplink

# Uso: compartir vía WhatsApp, Telegram, email, etc.
func share_controller_config():
    var link = create_shareable_droidpad_link("my_game", "192.168.1.100", 8080)
    OS.shell_open("https://wa.me/?text=%s" % link.uri_encode()) # WhatsApp

```

99.5. Formato de Eventos en Tiempo Real

99.5.1. Eventos Recibidos desde DroidPad

Cuando los usuarios interactúan con controles en la app DroidPad, se envían eventos al servidor en formato JSON.

Cada evento contiene: - **id**: Identificador único del elemento - **type**: Tipo de control - **Datos específicos**: Dependen del tipo

99.5.2. Tabla de Eventos por Tipo

99.5.2.1. BUTTON (Botón)

```
{
    "id": "btn_fire",
    "type": "BUTTON",
    "state": "PRESS"
}
```

99. DroidPad: Integración, Conexión y QR

Estados posibles: - PRESS - Botón presionado (descenso) - RELEASE - Botón liberado (ascenso) - CLICK - Un click completo (presion + liberación)

99.5.2.2. DPAD (Botones Direccionales)

```
{  
  "id": "arrow_keys",  
  "type": "DPAD",  
  "button": "RIGHT",  
  "state": "CLICK"  
}
```

Direcciones: LEFT, RIGHT, UP, DOWN Estados: PRESS, RELEASE, CLICK

99.5.2.3. JOYSTICK (Palanca Analógica)

```
{  
  "id": "left_stick",  
  "type": "JOYSTICK",  
  "x": 0.5,  
  "y": -0.75  
}
```

Importante: - Rango: **x y y de -1.0 a +1.0** (normalizado) - Centro: 0.0 - Sin campo “state”
- es solo la posición actual - Se envía continuamente mientras se mueve

Interpretación de ejes:

y = -1.0
↑
|
x = -1.0 ← 0,0 → x = +1.0
|
↓
y = +1.0

99.5.2.4. SLIDER (Control Deslizante)

```
{  
  "id": "volume",  
  "type": "SLIDER",  
  "value": 0.85  
}
```

- Rango: **Desde minValue a maxValue** definidos en la configuración
- Típicamente 0.0 a 1.0, pero completamente configurable
- Se envía continuamente

99.5.2.5. SWITCH (Interruptor ON/OFF)

```
{
  "id": "power",
  "type": "SWITCH",
  "state": true
}
```

- state: true (ON) o false (OFF)
- Se envía al cambiar de estado

99.5.2.6. STEERING_WHEEL (Volante de Dirección)

```
{
  "id": "steering",
  "type": "STEERING_WHEEL",
  "angle": 45.233445
}
```

- Rango: **-180 a +180 grados**
- 0° = posición neutra (recto)
- Positivo = rotación horaria (derecha)
- Negativo = rotación contra-horaria (izquierda)
- Se envía continuamente

99.5.2.7. ACCELEROMETER (Sensor de Aceleración)

```
{
  "type": "ACCELEROMETER",
  "x": 0.31892395,
  "y": -0.97802734,
  "z": 10.049896
}
```

- Unidad: **m/s²** (metros por segundo al cuadrado)
- Incluye gravedad (Z ~ 9.8 en reposo)
- Rango típico: -20 a +20 m/s²
- Ejes: X (lateral), Y (frontal), Z (vertical)

99.5.2.8. GYROSCOPE (Sensor Giroscópico)

```
{
  "type": "GYROSCOPE",
  "x": 0.15387291,
  "y": -0.22954187,
  "z": 0.08163925
}
```

99. DroidPad: Integración, Conexión y QR

- Unidad: **rad/s** (radianes por segundo)
 - Rango típico: - a + (-3.14159 a +3.14159)
 - Velocidad de rotación alrededor de cada eje
 - No incluye posición, solo velocidad de giro
-

99.6. Protocolo de Comunicación

99.6.1. Conexión TCP

Flujo de conexión:

1. Cliente (DroidPad) inicia conexión TCP
2. Se conecta a host:puerto especificado
3. Envía eventos JSON, uno por línea
4. Servidor procesa cada línea como evento independiente
5. Conexión se mantiene abierta hasta cierre

Formato:

```
{evento JSON 1}\n{evento JSON 2}\n{evento JSON 3}\n...
```

Ejemplo de stream TCP recibido:

```
{"id":"left_stick","type":"JOYSTICK","x":0.0,"y":0.0}\n{"id":"left_stick","type":"JOYSTICK","x":0.1,"y":0.0}\n{"id":"left_stick","type":"JOYSTICK","x":0.2,"y":-0.1}\n{"id":"btn_a","type":"BUTTON","state":"PRESS"}\n{"id":"btn_a","type":"BUTTON","state":"RELEASE"}
```

99.6.2. Conexión WebSocket

Similar a TCP, pero con protocolo WebSocket (ws:// o wss://):

```
// Cliente WebSocket\nconst ws = new WebSocket("ws://192.168.1.100:8080");\n\nws.onmessage = (event) => {\n    const data = JSON.parse(event.data);\n    console.log("Evento recibido:", data);\n};
```

99.6.3. MQTT

Para usar MQTT, DroidPad envía eventos a un tópico específico:

Tópico: droidpad/{pad_name}/{element_id}
 Payload: {"type": "JOYSTICK", "x": 0.5, "y": -0.75}

99.6.4. UDP

Eventos enviados como datagramas UDP:

Destino: {host}:{port}
 Payload: {evento JSON}

Nota: UDP no garantiza entrega, ideal para datos de alta frecuencia donde la pérdida ocasional es aceptable.

99.7. Preguntas Frecuentes

99.7.1. P1: ¿Puedo usar diferentes protocolos simultáneamente?

R: No en una misma conexión de pad. Debes crear pads separados para cada protocolo.

Pad 1 (TCP) - Conectado a 192.168.1.100:8080
 Pad 2 (WebSocket) - Conectado a 192.168.1.50:9000
 Pad 3 (MQTT) - Conectado a broker.example.com:1883

99.7.2. P2: ¿Qué tan frecuentes son los eventos?

R: Depende del tipo de control: - **Joystick/Slider/Steering:** ~60 Hz (cada ~16ms) mientras se mueven - **Button/DPAD/Switch:** Solo al cambiar estado - **Accelerometer/Gyroscope:** 30-60 Hz según sensor

99.7.3. P3: ¿La precisión es suficiente para juegos?

R: Sí. Los joysticks tienen precisión de float (6-8 decimales). Es comparable a controles físicos profesionales.

99.7.4. P4: ¿Puedo modificar el QR después de generararlo?

R: El QR es estático - encadena datos específicos. Para cambiar la configuración, debes: 1. Generar un nuevo QR 2. O cargar la configuración vía archivo JSON

99.7.5. P5: ¿Qué sucede si el servidor está offline cuando se escanea el QR?

R: DroidPad intentará conectar según el `timeout` especificado. Si falla: - Reintentar automáticamente - Mostrar error al usuario - Permitir configuración manual

99.7.6. P6: ¿Puedo usar localhost (127.0.0.1)?

R: No. El teléfono es un dispositivo diferente, localhost refiere a sí mismo. Usa: - IP local de tu máquina (ej: 192.168.1.x) - IP pública (si están en redes diferentes) - Hostname con DNS resuelto

99.7.7. P7: ¿El protocolo es seguro?

R: No tiene autenticación en los protocolos base. Para producción: - Usa `wss://` (WebSocket seguro) en lugar de `ws://` - Implementa verificación de token/API key - Usa VPN o red privada - No expongas puertos públicos directamente

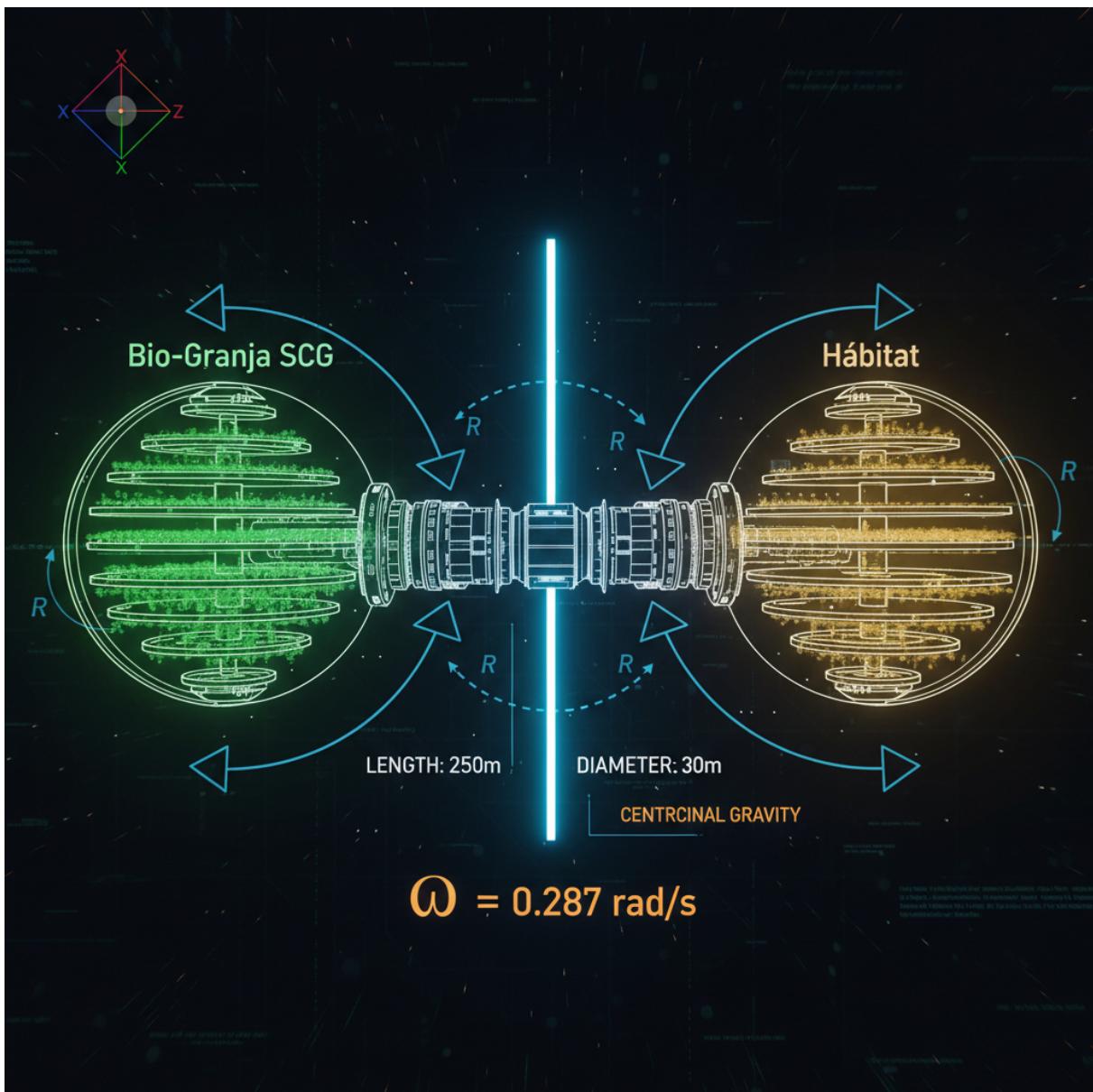
99.7.8. P8: ¿Puedo enviar comandos de vuelta al teléfono?

R: Depende del protocolo: - **TCP/WebSocket:** Sí, es bidireccional. Envía JSON y DroidPad puede procesarlo - **UDP:** No, es unidireccional (móvil → servidor) - **MQTT:** Sí, usa tópicos de respuesta

99.8. Referencias

- GitHub: [UmerCodez/DroidPad](#)
- QR Server API - Generador QR gratis
- Deep Linking
- Licencia: GPL-3.0

100. Física del SCG (Sistema Centrífugo de Gravedad) - Odisea



He preparado una **documentación completa y detallada** sobre la implementación de los cálculos físicos del SCG para Godot 3. Aquí está el resumen ejecutivo:

100.0.1. Archivos Generados

1. `odisea_scg_physics.md` - Documento teórico exhaustivo con:

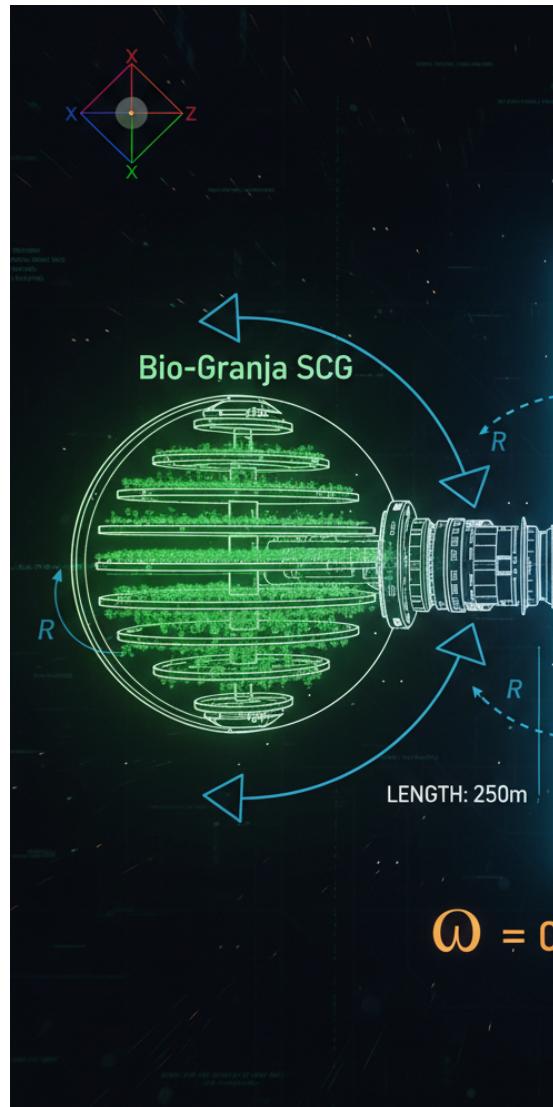
- Fundamentos matemáticos de gravedad centrífuga

100. Física del SCG (Sistema Centrífugo de Gravedad) - Odisea

- Cálculos geométricos para plataformas cóncavas
- Fórmulas para esferas conectadas con tubos
- Interpolación de gravedad en transiciones
- Ejemplos prácticos para Odisea
-

2. `godot3_scg_implementation.gd` - Código funcional GDScript con:

- Clase `GravitySource` completa
- `PlayerController` modificado con gravedad variable
- Sistema de zonas (`GravityZone`)
- Plataformas rotantes (`RotatingPlatform`)
- Cache de rendimiento



3. `scg_reference_quick.md` - Guía rápida con tablas y fórmulas
Fórmulas Clave

Aceleración centrífuga:

$$a = \omega^2 \cdot r$$

Para 1G a r=100m:

$$\omega = 0.287 \text{ rad/s}, \quad T = 21.9 \text{ seg}, \quad v = 34.4 \text{ m/s}$$

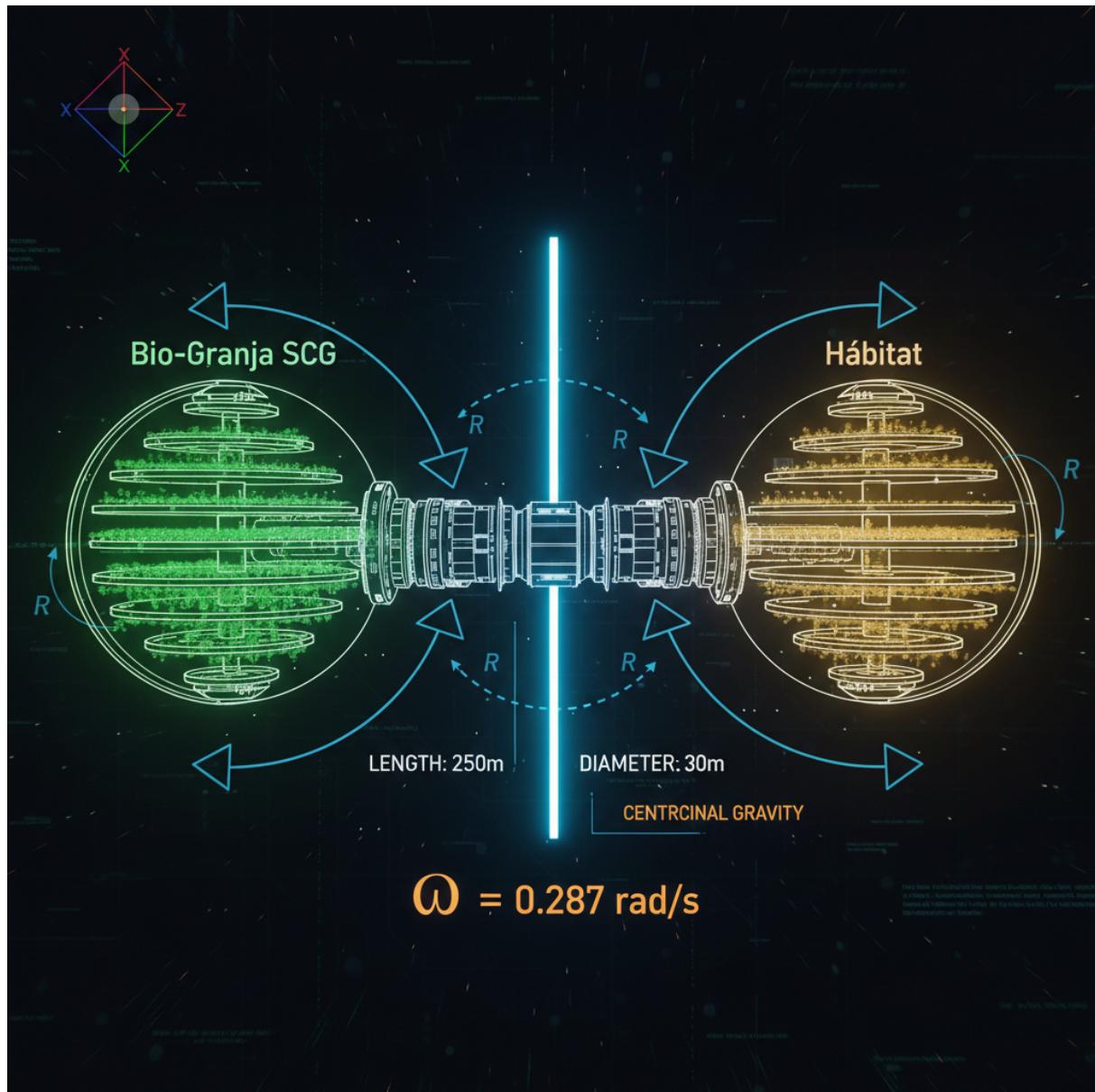


Figure 100.1.: SCG dual-sphere rotating gravity system with centrifugal force vectors

SCG dual-sphere rotating gravity system with centrifugal force vectors

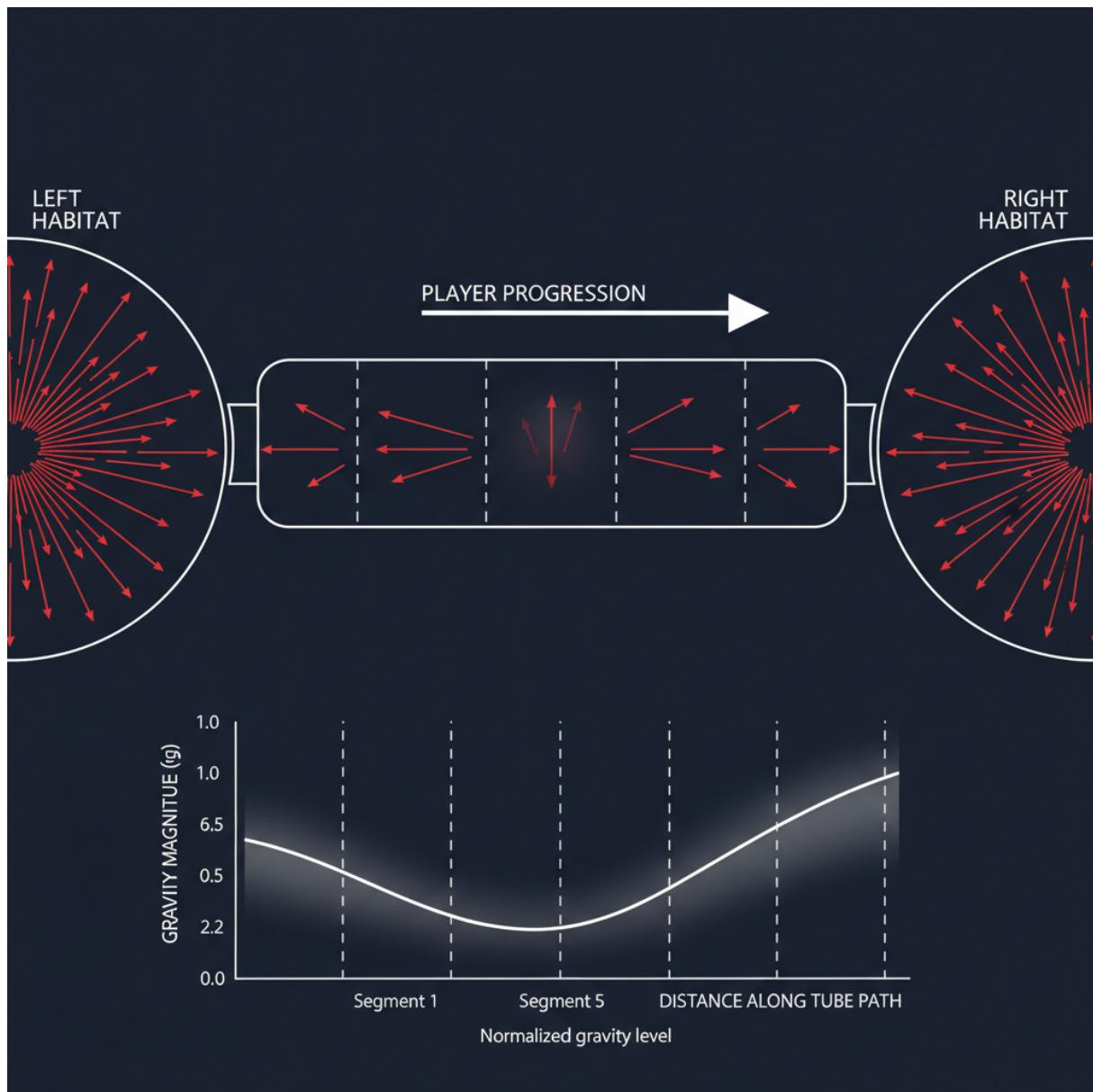


Figure 100.2.: Gravity interpolation through tube connector - transitional field visualization

Gravity interpolation through tube connector - transitional field visualization

100.0.2. Configuración Propuesta para Odisea

Dos esferas conectadas:

Componente	Esfera A (Bio-Granja)	Esfera B (Hábitat)	Tubo
Radio	120 m 0.287 rad/s	100 m 0.287 rad/s	310 m largo Interpolado
Gravedad	1.0G	0.82G	Transición suave
Período	21.9 seg	21.9 seg	-

Interpolación en tubo:

```
Gravity(s) = lerp(g_A, g_B, smooth_step(s))
donde s [0, 1] es la posición en el tubo
```

100.0.3. Implementación Godot 3

Cálculo de gravedad en punto:

```
func get_gravity(point: Vector3, omega: float) -> Vector3:
    var perp_dist = sqrt(point.x*point.x + point.z*point.z)
    var g_mag = omega * omega * perp_dist
    var g_dir = -Vector3(point.x, 0, point.z) / perp_dist
    return g_dir * g_mag
```

Alineación del jugador:

- Rotación suave del personaje conforme cambia la dirección de “abajo”
- Salto contra la dirección de gravedad actual
- Movimiento en plano tangente a la gravedad

100.0.4. Casos de Uso en Odisea

Acto II - Bio-Granjas SCG:

- Plataformas cóncavas que parecen suelos pero están sobre esferas rotantes
- **Puzzle:** Realinear brazos giratorios para cambiar de una esfera
- **Efecto visual:** Paredes se convierten en suelos cuando gira

Área Tubular:

- Transición progresiva de gravedad mientras se cruza el tubo
- Cambio de orientación de “arriba/abajo”
- Desafío: mantener movimiento mientras la gravedad rota

100.0.5. Optimizaciones

- **Caching:** Almacenar gravedad por posición redondeada
- **LOD:** Desactivar cálculos complejos a distancia > 200m
- **Raycast volumétrico:** Detectar zona actual del jugador

100.0.6. Visualizaciones Generadas

- Diagrama de sistema dual-esfera con vectores de gravedad

