UNIVERSIDAD CATÓLICA DE CUENCA

UNIVERSIDAD CATÓLICA DE CUENCA

Comunidad Educativa al Servicio del Pueblo

FACULTAD DE INFORMÁTICA, CIENCIAS DE LA COMPUTACIÓN E INNOVACIÓN TECNOLÓGICA

CARRERA DE REALIDAD VIRTUAL Y VIDEOJUEGO

Videojuego educativo basado en gimnasia cerebral para el desarrollo de habilidades cognitivas en niños de 4 a 5 años.

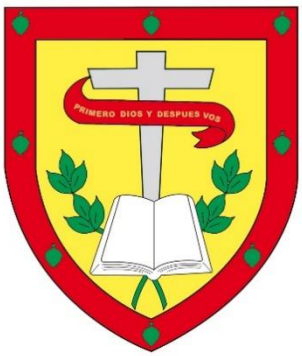
PROYECTO DE INTEGRACIÓN CURRICULAR PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN REALIDAD VIRTUAL Y VIDEOJUEGOS

AUTOR: CARLOS BOLIVAR MALDONADO MOSCOSO
DIRECTOR: ING. JOHN LUIS ESTRADA GUAYTA, M.S.

CUENCA - ECUADOR

2026

DIOS, PATRIA, CULTURA Y DESARROLLO



UNIVERSIDAD CATÓLICA DE CUENCA

Comunidad Educativa al Servicio del Pueblo

FACULTAD DE INFORMÁTICA, CIENCIAS DE LA COMPUTACIÓN E INNOVACIÓN TECNOLÓGICA

CARRERA DE REALIDAD VIRTUAL Y VIDEOJUEGO

Videojuego educativo basado en gimnasia cerebral para el desarrollo de habilidades cognitivas en niños de 4 a 5 años

PROYECTO DE INTEGRACIÓN CURRICULAR PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN REALIDAD VIRTUAL Y VIDEOJUEGOS

AUTOR: CARLOS BOLIVAR MALDONADO MOSCOSO
DIRECTOR: ING. JOHN LUIS ESTRADA GUAYTA, M.S.

CUENCA - ECUADOR

2026

DIOS, PATRIA, CULTURA Y DESARROLLO

Universidad Católica de Cuenca

DECLARATORIA DE AUTORÍA Y RESPONSABILIDAD

Declaratoria de Autoría y Responsabilidad

Carlos Bolivar Maldonado Moscoso portador de la cédula de ciudadanía Nº **0150047462**. Declaro ser el autor de la obra: "**Videojuego educativo basado en gimnasia cerebral para el desarrollo de habilidades cognitivas en niños de 4 a 5 años**", sobre la cual me hago responsable sobre las opiniones, versiones e ideas expresadas. Declaro que la misma ha sido elaborada respetando los derechos de propiedad intelectual de terceros y eximo a la Universidad Católica de Cuenca sobre cualquier reclamación que pudiera existir al respecto. Declaro finalmente que mi obra ha sido realizada cumpliendo con todos los requisitos legales, éticos y bioéticos de investigación, que la misma no incumple con la normativa nacional e internacional en el área específica de investigación, sobre la que también me responsabilizo y eximo a la Universidad Católica de Cuenca de toda reclamación al respecto.

Cuenca, 08/09/2026

F:.....

Carlos Bolivar Maldonado Moscoso

C.I. **0150047462**

Universidad Católica de Cuenca

CERTIFICADO

Certificado

Certifico que la presente Propuesta Tecnológica fue desarrollado por **Carlos Bolivar Maldonado Moscoso** portador(a) de la cedula de ciudadanía N.º **0150047462** con el tema "**Videojuego educativo basado en gimnasia cerebral para el desarrollo de habilidades cognitivas en niños de 4 a 5 años**", bajo mi supervisión.

Cuenca, **08 de septiembre de 2026**

F: .....

Ing. John Luis Estrada Guayta, M.S.

Docente - Tutor

Agradecimiento

Agradezco especialmente a mi tutor, John, quien durante este proceso ha sido mucho más que un tutor, convirtiéndose también en un amigo y una guía importante. Gracias por enseñarme que siempre hay que investigar, buscar más allá de lo evidente y esforzarse por sorprender con aquello a lo que uno se dedica. También por recordarme la importancia de ser creativo y de intentar siempre hacer las cosas de una manera diferente y mejor.

A mis amigos, gracias por todas las experiencias y conocimientos compartidos. De cada uno aprendí cosas que antes no sabía y que, de una u otra manera, me ayudaron a crecer tanto personal como profesionalmente. Muchas de esas enseñanzas forman parte de lo que hoy me ha permitido llegar hasta este momento.

Quiero recordar también a mi abuelo. No pudo verme graduarme del colegio y hoy tampoco puede estar presente para verme culminar esta etapa universitaria, pero espero que, donde quiera que esté, se sienta orgulloso de mí y de todo lo que he logrado.

Finalmente, agradezco profundamente a mis padres, quienes han estado presentes durante todo este camino. Espero que este logro también sea motivo de orgullo para ellos, porque gran parte de lo que hoy soy y de lo que he conseguido ha sido posible gracias a su apoyo, esfuerzo y enseñanzas.

A todos quienes formaron parte de este proceso, gracias por haber aportado de alguna manera para que hoy pueda llegar hasta aquí.

Dedicatoria

Dedico este trabajo a mis padres.

A mi madre, por aguantarme durante todo este tiempo, por darme un techo, fuerzas y nunca dejar que me faltara un plato de comida en la mesa. Gracias por estar presente y apoyarme durante todo este camino.

A mi padre, por enseñarme lo que significa el esfuerzo y la convicción de salir adelante, por más cansado o pesado que sea el camino. Me enseñó que, si realmente se quiere algo, hay que hacerlo, pero hacerlo bien, porque un trabajo bien hecho siempre termina siendo reconocido.

A mi abuelo, quien, a pesar de llevar cuatro años sin estar entre nosotros, sigue presente en muchas de las cosas que me enseñó. Me enseñó que incluso lo malo puede tener un lado divertido, que a veces hay que dejar lo que nos gusta para concentrarnos en lo que debemos hacer, pero sin perder nunca nuestra esencia. También me enseñó a ser sencillo, sin importar qué tan alto se pueda llegar, a ayudar sin pensarlo demasiado y a confiar en que cosas mejores vendrán de la mano de Dios. Sobre todo, me enseñó a ser cariñoso y a recordar que somos humanos y que nadie es perfecto.

Por último, a mi enamorada. Gracias por todo el tiempo que hemos compartido, por enseñarme a no rendirme y por no dejarme caer cuando ya no sabía qué hacer. Gracias por recordarme que todo puede salir bien siempre que pongamos de nuestra parte y confiemos en Dios, y que a veces las cosas pasan por alguna razón.

Gracias por estar conmigo en este camino. Los amo.

Resumen

El presente trabajo de titulación tuvo como objetivo desarrollar un prototipo funcional de videojuego educativo 2D denominado Mundo Aprendo, implementado en Unity y orientado a niños de 4 a 5 años. El proyecto surgió de la necesidad de disponer de recursos digitales organizados, comprensibles y adaptados a este grupo etario, debido a que algunas aplicaciones infantiles presentan actividades aisladas, instrucciones poco claras y sistemas de progresión limitados. El videojuego integra cuatro mundos temáticos y cuatro minijuegos principales: Mundo Musical, basado en la interacción con teclas de piano y la repetición de secuencias; Mundo de los Cuentos, orientado a la selección de relatos y el reconocimiento de voz; Mundo de los Tamaños, centrado en la clasificación de animales grandes y pequeños; y Mundo de las Emociones, destinado a identificar emociones mediante imágenes y situaciones. El desarrollo siguió una adaptación operativa basada en Kanban, organizando las actividades en un flujo continuo de tareas que abarcó la producción de recursos gráficos, la construcción de escenas e interfaces, la programación de mecánicas, la integración de los sistemas complementarios y la comprobación funcional. Las pruebas funcionales internas permitieron verificar la navegación entre escenas, la selección de objetos, el reconocimiento de voz y la retroalimentación mediante sonidos, animaciones y estrellas. El proyecto no incluyó evaluación pedagógica ni pruebas con niños, por lo que sus resultados se limitan al funcionamiento técnico del prototipo.

Palabras clave: *Videojuego educativo, gimnasia cerebral, educación infantil, Unity, habilidades cognitivas.*

Abstract

This undergraduate project aimed to develop a functional 2D educational video game prototype entitled Mundo Aprendo, built in Unity and designed for children aged four to five. The project addressed the need for digital learning resources that are structured, understandable, and appropriate for this age group, since many children's applications rely on isolated activities, unclear instructions, and limited progression systems. The game includes four themed worlds and four main mini-games: Musical World, focused on piano-key interaction and sequence repetition; Story World, centered on story selection and voice recognition; Size World, based on identifying large and small animals; and Emotion World, designed to recognize emotions through images and contextual situations. Development followed an operational adaptation based on Kanban, organizing the work as a continuous task flow covering graphic asset production, scene and interface construction, gameplay programming, integration of complementary systems, and functional verification. Internal functional tests verified scene navigation, object selection, voice-recognition features, and feedback mechanisms based on sound effects, animations, and star rewards. The project did not include pedagogical assessment, cognitive-skill measurement, or testing with children; therefore, the reported findings are limited to the technical implementation and functional performance of the prototype.

Keywords: *Educational video game, brain training, early childhood education, Unity, cognitive skills.*

Índice

Índice de tablas

Índice de ilustraciones

Capítulo I

1. Marco Referencial

Este capítulo presenta el contexto, el problema y la orientación técnica de Mundo Aprendo. El proyecto desarrolla un videojuego educativo 2D para PC Windows, dirigido a niños de 4 a 5 años. También define el alcance, los módulos, el método de trabajo y las pruebas funcionales previstas.

En los siguientes apartados se explican el problema identificado, los objetivos, el alcance, la metodología, los beneficiarios y los resultados esperados del proyecto.

1.1 Contextualización del problema

El proyecto se orienta al desarrollo de un videojuego educativo 2D para PC Windows, dirigido a niños de 4 a 5 años, grupo de edad definido dentro del planteamiento pedagógico del proyecto. Para ese rango, UNESCO y UNICEF señalan que el uso de tecnología digital depende del contenido, del contexto y de la forma de interacción [1], [2]. Ese criterio permite caracterizar el contexto digital en el que se inserta la propuesta y orienta decisiones técnicas como la brevedad de las consignas, el tipo de entrada y la retroalimentación inmediata.

Mundo Aprendo integra personajes, escenarios, mundos y minijuegos cuya interacción es visual y directa: la respuesta se da con un clic sobre el elemento que aparece en pantalla —una tecla, un animal o una emoción—, sin menús intermedios, sin arrastre y sin comandos escritos. La interfaz usa instrucciones breves, retroalimentación audiovisual y un avance organizado entre actividades.

Mundo Aprendo es un videojuego educativo 2D pensado para funcionar en computadoras con Windows. El niño usará principalmente el mouse para hacer clic y

seleccionar elementos. El juego incluirá actividades de secuencias musicales, lectura apoyada por voz, comparación de tamaños y reconocimiento de emociones, organizadas mediante mundos y niveles.

1.2 Planteamiento del problema

Actualmente, los recursos digitales dirigidos a niños de 4 a 5 años se distribuyen entre aplicaciones móviles, entornos de escritorio y soluciones que incorporan cámaras, sensores o entornos de realidad virtual, y cada una de esas configuraciones impone requisitos propios de instalación, calibración y comprobación [3]. Las revisiones sobre aprendizaje basado en juegos y sobre juego digital en los primeros años coinciden en que la interactividad, la adecuación a la edad, el contexto de uso y la facilidad de acción son los criterios que determinan si un recurso resulta utilizable en ese rango de edad [4], [5]. Ese conjunto de condiciones delimita el escenario en el que se sitúa el problema.

En ese escenario se presentan dificultades relacionadas con la integración de los componentes de una experiencia digital infantil. Parte de las aplicaciones disponibles ofrece actividades que funcionan de manera aislada, con instrucciones que no siempre resultan comprensibles para la edad y con sistemas de progresión limitados, de modo que el conjunto no llega a comportarse como una experiencia única. La dificultad no reside en la existencia de las actividades, sino en la ausencia de una estructura que las relacione mediante una navegación común, una forma estable de responder y un registro del avance que se conserve entre sesiones.

Estas dificultades se deben a causas de orden técnico que pueden identificarse con precisión. La primera es que el dispositivo de entrada condiciona la forma en que el sistema recibe y valida una acción: un exergame para niños de 4 a 5 años convirtió el movimiento

corporal en entrada del juego, mientras que otro proyecto preescolar organizó su diseño mediante etapas iterativas con participación de especialistas, docentes y niños [6], [7]. Ambos casos muestran que la entrada y la retroalimentación deben quedar definidas desde la planificación y no resolverse una vez construidas las actividades. La segunda causa es la dispersión de plataformas y periféricos: cuando una propuesta depende de cámaras, sensores o equipos adicionales, aumenta el número de elementos que deben configurarse y comprobarse antes de que la actividad llegue siquiera a ejecutarse [3].

Esto genera consecuencias observables tanto en el uso como en el desarrollo. Del lado del uso, una integración deficiente produce navegación poco intuitiva, mecánicas desconectadas entre sí y actividades sin relación aparente, de modo que quien utiliza el recurso percibe un conjunto de pantallas independientes en lugar de un recorrido continuo. Del lado del desarrollo, la falta de una organización común de escenas, controles y persistencia obliga a que cada actividad resuelva por su cuenta la validación, el sonido y el registro del avance, lo que multiplica el trabajo de comprobación y dificulta localizar el origen de un fallo cuando el conjunto deja de comportarse como se espera.

1.3 Formulación del problema

¿Cómo desarrollar un prototipo funcional de videojuego educativo 2D en Unity que integre personajes, escenarios, mundos y minijuegos para niños de 4 a 5 años?

De la pregunta principal se derivan tres preguntas complementarias que orientaron el trabajo técnico y que se responden a lo largo del documento:

¿Qué recursos gráficos, escenas e interacciones requiere el prototipo para integrar cuatro actividades temáticas dirigidas a niños de 4 a 5 años?

¿Cómo puede organizarse la navegación, la retroalimentación y el progreso para mantener un flujo técnico coherente entre los mundos y sus minijuegos?

¿Qué pruebas funcionales permiten comprobar la navegación, la interacción, la persistencia y el módulo de voz dentro del entorno Windows definido?

1.4 Justificación

En el plano académico, el proyecto se justifica porque reúne en un único producto verificable las competencias que la carrera desarrolla por separado a lo largo de la formación: la producción de recursos gráficos 2D, la programación en C# dentro de Unity, el diseño e integración de interfaces, la organización de escenas y la compilación final para una plataforma concreta. Un trabajo que atraviesa esas fases obliga a sostener decisiones técnicas coherentes entre sí durante todo el desarrollo, y no solo a resolver ejercicios aislados, por lo que su valor formativo reside tanto en el prototipo entregado como en la documentación del proceso que lo produjo.

En el plano tecnológico, la propuesta se justifica porque integra dentro de un solo ejecutable recurso 2D, escenas, navegación, entradas del usuario, audio, progreso local y una compilación para Windows. La conveniencia de separar la producción de recursos, la programación de mecánicas y la integración antes de reunir los minijuegos en un mismo flujo se respalda en un antecedente que desarrolló un juego educativo en Unity 2D y documentó su proceso de construcción [8]. Del mismo modo, la traducción de una actividad al medio digital mediante una consigna, una entrada del usuario, una validación y una respuesta se apoya en la forma en que las guías revisadas organizan las actividades a partir de una descripción, unos recursos, unos movimientos y una secuencia de aplicación [9], [10].

En el plano social e institucional, el prototipo aporta un producto funcional y documentado que puede utilizarse como material de referencia para trabajos posteriores de la carrera, tanto en la parte artística como en la de programación. Su ejecución es local y no depende de servidores externos ni de bases de datos remotas, condición que reduce las exigencias de instalación y administración en contextos de aula o de hogar. Conviene precisar el alcance de esta justificación: el aporte se sitúa en la disponibilidad de un recurso técnico documentado, y no en un beneficio educativo comprobado, que exigiría un estudio posterior con el público objetivo.

En el plano económico y operativo, la solución se justifica porque funciona sobre equipos con Windows de los que las instituciones ya disponen y no introduce costos recurrentes de infraestructura: al no emplear base de datos ni servicios remotos, no existe mantenimiento de servidores ni administración de cuentas de usuario. La distribución consiste en trasladar completa la carpeta de compilación, y la verificación de que un recurso cumple las condiciones de uso antes de incorporarlo sigue el criterio de un antecedente que evaluó aspectos pedagógicos, de usabilidad, de seguridad y de calidad antes de seleccionar aplicaciones para Inicial II [11]. Ese mismo criterio ordena las pantallas, documenta las condiciones de ejecución y revisa cada función antes de darla por incorporada.

En el plano investigativo, el trabajo deja una base técnica documentada sobre la que pueden apoyarse estudios posteriores. El proyecto registra la arquitectura de clases, la organización de escenas, las claves de persistencia, las pruebas ejecutadas y las evidencias asociadas, de manera que una evaluación futura con docentes, cuidadores o niños partiría de un artefacto verificado y no de una versión indeterminada. Delimitar con precisión qué

se comprobó y qué no forma parte de ese aporte: el prototipo entrega comportamiento de software verificable, y las preguntas sobre usabilidad, comprensión o efectos de aprendizaje quedan planteadas como líneas de investigación con su propio diseño metodológico.

1.5 Objetivo general

Desarrollar un prototipo funcional de videojuego educativo 2D denominado Mundo Aprendo en Unity, mediante el diseño e integración de personajes y entornos visuales y la programación de minijuegos con mecánicas interactivas, orientado a niños de 4 a 5 años.

1.6 Objetivos específicos

- Crear los recursos gráficos 2D del videojuego, incluyendo personajes, entornos y elementos visuales requeridos para los mundos y minijuegos del proyecto.
- Diseñar la estructura de escenas, mundos y niveles del videojuego en Unity 2D, estableciendo una navegación funcional y coherente entre sus diferentes módulos.
- Programar en Unity 2D las mecánicas de interacción y la lógica de funcionamiento de los minijuegos, considerando acciones de clic y selección directa, validaciones y retroalimentación.
- Implementar mediante programación los sistemas complementarios del prototipo, incluidos el control de avance, el sistema de estrellas, la activación visual de planetas y el reconocimiento de voz del módulo correspondiente.
- Evaluar el funcionamiento técnico del videojuego mediante pruebas funcionales de navegación, interacción, progresión y respuesta de los minijuegos, para verificar su ejecución correcta en PC Windows bajo las condiciones documentadas.

1.7 Alcance del proyecto

El proyecto abarcó el desarrollo y la comprobación técnica de un prototipo funcional de videojuego educativo 2D en Unity para computadoras con Windows. Los niños de 4 a 5 años constituyen el público de diseño del producto, no una muestra de investigación. La versión implementada se utiliza principalmente con el mouse e incluye una actividad de lectura apoyada por reconocimiento de voz.

La versión implementada incluye los siguientes elementos:

- Personajes, fondos, objetos interactivos, elementos de interfaz y animaciones 2D simples.
- Menú principal, navegación entre mundos o planetas, niveles y escenas de actividad.
- Minijuegos de secuencias musicales, lectura apoyada por voz, comparación de tamaños y reconocimiento de emociones.
- Mecánicas de clic y selección directa, validaciones, mensajes de retroalimentación, sonidos y respuesta visual.
- Sistema de progreso, estrellas y activación visual de mundos conforme se completan actividades.

El proyecto no incluyó publicación comercial, versiones móvil o web, servicios en línea permanentes, bases de datos externas, multijugador ni pagos. Su aporte consistió en materializar la idea y comprobar técnicamente el prototipo en PC Windows mediante pruebas internas automatizadas y recorridos funcionales documentados. No se realizaron sesiones con niños, evaluación pedagógica ni mediciones de usabilidad, comprensión, aceptación, aprendizaje o efectividad educativa.

1.8 Delimitaciones

A diferencia del alcance, que enumera los componentes desarrollados, las delimitaciones precisan las condiciones bajo las cuales se construyó y verificó el prototipo, así como los aspectos que no fueron investigados. La Tabla 1 presenta los límites temáticos, poblacionales, tecnológicos, funcionales, espaciales, temporales y de verificación.

Tabla 1

Delimitaciones del proyecto Mundo Aprendo.

Dimensión	Límite y exclusiones
Temática	Desarrollo, integración y documentación técnica de un videojuego educativo 2D. No comprende una intervención pedagógica ni la comprobación de efectos educativos.
Poblacional	Los niños de 4 a 5 años constituyen el público de diseño. No se conformó una muestra humana ni se recopilaron datos de niños, docentes o cuidadores.
Tecnológica	Unity 2D, PC Windows x86-64, mouse, persistencia local y voz dependiente de Windows. Se excluyen versiones móviles y web, servidores, bases remotas, multijugador y servicios permanentes en línea.
Funcional	Cuatro mundos, sus mecánicas, retroalimentación, estrellas y progreso. No incluye publicación comercial ni ampliación de contenidos fuera de la versión documentada.
Espacial	Comprobación en el entorno local de desarrollo y ejecución disponible. No se realizó despliegue institucional ni aplicación en aula.
Temporal	Resultados limitados a la versión y a las ejecuciones registradas durante el periodo de titulación; no existe seguimiento longitudinal.
Verificación	Pruebas internas EditMode, PlayMode, voz, compilación y comprobación funcional. Se excluyen usabilidad, comprensión, aceptación, aprendizaje y efectos cognitivos con el público objetivo.

Nota. Elaboración propia con base en el alcance y en la evidencia técnica del proyecto.

1.9 Beneficiarios

Los beneficiarios se distinguen según usen el juego directamente o acompañen al niño durante la actividad. Mundo Aprendo se plantea como una experiencia lúdica digital con potencial para complementar el acompañamiento de docentes, familias y cuidadores.

Tabla 2

Beneficiarios del proyecto.

Tipo	Beneficiarios	Aporte previsto
Directos	Niños de 4 a 5 años.	Experiencia de juego con actividades visuales, mundos y minijuegos organizados.
Indirectos	Docentes de educación inicial.	Recurso lúdico digital con potencial para complementar actividades de acompañamiento.
Indirectos	Padres, madres y cuidadores.	Recurso para acompañar el uso del videojuego y sus actividades.
Indirectos	Comunidad académica.	Referencia sobre el desarrollo técnico de un videojuego educativo infantil.

Nota. Elaboración propia.

1.10 Metodología general

El proyecto se abordó como una propuesta tecnológica de carácter aplicado. La revisión documental permitió identificar criterios de diseño, de interacción y de organización de actividades, mientras que la inspección directa del proyecto de Unity y de sus registros permitió describir la implementación existente. Los resultados se expresan mediante requerimientos, componentes, capturas y pruebas funcionales, sin realizar inferencias sobre aprendizaje o desarrollo cognitivo.

El trabajo es descriptivo y de desarrollo tecnológico. Es descriptivo porque documenta escenas, controles, persistencia, servicios y comportamiento observable; y es tecnológico porque organiza esos hallazgos en un prototipo ejecutable. No corresponde a un diseño experimental con usuarios, dado que no se conformaron grupos, no se aplicaron tratamientos y no se midieron variables educativas.

El desarrollo de Mundo Aprendo se organizó mediante una adaptación operativa basada en Kanban, un enfoque de organización del trabajo basado en la visualización de las tareas y en su avance mediante un flujo continuo. Esta elección respondió a las condiciones reales del proyecto: un desarrollo realizado por una sola persona, con actividades de naturaleza heterogénea —producción artística, programación, integración y comprobación— cuya duración era difícil de estimar de antemano. A diferencia de los enfoques basados en iteraciones temporales cerradas, Kanban no exige agrupar el trabajo en ciclos de duración fija, por lo que cada tarea avanzó según su propio estado de desarrollo y verificación.

Las tareas se representaron como unidades independientes de trabajo que cambiaban de estado conforme se desarrollaban, se revisaban y se daban por terminadas. Su

incorporación fue progresiva: cada nueva necesidad detectada durante la construcción del prototipo se incorporó como una tarea más al flujo, sin reabrir una planificación global. La priorización siguió un criterio de dependencia técnica, de modo que las tareas que habilitaban otras posteriores se atendieron primero; así, los recursos gráficos y la estructura de escenas precedieron a la programación de mecánicas, y esta precedió a la integración de los sistemas de progreso, estrellas, audio y voz. Este orden permitió que cada elemento se comprobara antes de que otro dependiera de él y que las incidencias se localizaran de forma temprana.

Los grupos de tareas gestionados dentro del flujo correspondieron a la producción de recursos gráficos 2D, el diseño y construcción de interfaces, la creación y configuración de escenas, el desarrollo de las mecánicas de los minijuegos, la programación de los sistemas complementarios, la integración de los sistemas, las pruebas funcionales, la corrección de incidencias, la compilación para Windows y la documentación del proyecto. La Tabla 3 sintetiza esos grupos de trabajo y la evidencia asociada a cada uno. El flujo basado en Kanban permitió mantener visible el estado del conjunto en todo momento, lo que resultó especialmente útil en un desarrollo unipersonal, donde no existen reuniones de coordinación que informen del avance y la visualización del propio flujo cumple esa función.

El flujo de trabajo basado en Kanban se utilizó como mecanismo operativo para organizar y priorizar el desarrollo, pero no como instrumento de medición del proceso. Por esta razón, no se reportan métricas como número de tarjetas, límites de trabajo en curso, tiempo de ciclo o rendimiento del tablero, ya que no se conservó un registro histórico de esos valores. La metodología se documenta a partir de la forma en que las tareas fueron

incorporadas, priorizadas y cerradas conforme el proyecto requería recursos artísticos, escenas, programación, integración y comprobación funcional.

Tabla 3

Grupos de tareas gestionados en el flujo de trabajo del prototipo.

Grupo de tareas	Actividad principal	Evidencia documentada
Análisis técnico	Revisar los requerimientos aprobados y definir la implementación de escenas, mundos, minijuegos, recursos y sistemas funcionales.	Lista de requerimientos, pantallas, actividades y orden de desarrollo.
Producción de recursos gráficos	Preparar personajes, fondos, botones, objetos y animaciones sencillas en 2D.	Recursos gráficos organizados por mundo y actividad.
Construcción de escenas e interfaces	Montar el menú, las pantallas, la navegación y la estructura base del videojuego.	Escenas base y navegación funcional entre pantallas.
Programación e integración	Programar e integrar minijuegos, interacciones, validaciones, retroalimentación, avance, estrellas y lectura apoyada por voz.	Minijuegos y sistemas complementarios implementados e integrados al prototipo.
Pruebas funcionales y corrección	Comprobar navegación, acciones, progresión y respuesta de cada actividad en PC Windows.	Registro de pruebas, correcciones y prototipo ejecutable para verificación interna.

Nota. Elaboración propia.

La comprobación se realizó mediante pruebas funcionales internas sobre la apertura y transición de escenas, la respuesta de los controles, la ejecución de las reglas de cada minijuego, la persistencia del avance y el comportamiento del módulo de voz. Estas pruebas corresponden a verificación técnica del prototipo y no a una evaluación con usuarios; su detalle y su estado se presentan en el Capítulo IV.

Las condiciones de viabilidad del proyecto se describen a continuación, ya que determinaron el entorno en el que se aplicó esta metodología.

La propuesta fue viable en el entorno disponible porque se desarrolló con Unity 2D para PC Windows y reutilizó recursos locales del proyecto. La ejecución no depende de servidores externos, salvo el servicio de dictado de Windows empleado por el módulo de voz. En consecuencia, la viabilidad técnica se limita a la configuración documentada y requiere comprobar permisos, micrófono y compatibilidad antes de cada instalación.

La viabilidad operativa se relaciona con una interacción basada principalmente en el mouse y con la supervisión inicial de una persona adulta. No se efectuó una evaluación económica formal ni una validación con niños; por ello, el documento describe el funcionamiento técnico del prototipo y deja la medición de usabilidad, aceptación y efectos educativos para una fase posterior.

1.11 Resultados esperados

Al inicio del proyecto se establecieron como resultados esperados un conjunto de productos verificables, cada uno asociado con un objetivo específico. Este apartado conserva la redacción prospectiva de la planificación original; el estado realmente alcanzado por cada producto se documenta en los Capítulos III y IV y se valora en las conclusiones.

El primer resultado esperado era un conjunto de recursos gráficos 2D propios, compuesto por personajes, escenarios, objetos interactivos y elementos de interfaz, organizado por mundo y listo para su importación en Unity. El segundo era una estructura de escenas, mundos y niveles organizada en Unity 2D, con una navegación coherente entre el menú principal, la selección de mundos, los tutoriales, las actividades y el retorno.

El tercer resultado esperado era un conjunto de mecánicas programadas en C# que respondieran a acciones de clic y selección directa, validaran la respuesta del usuario y devolvieran retroalimentación visual y sonora. El cuarto correspondía a la implementación mediante programación de los sistemas complementarios: control de avance, sistema de estrellas, activación visual de planetas, persistencia local y lectura apoyada por reconocimiento de voz.

El quinto resultado esperado era un registro de pruebas funcionales que permitiera comprobar la navegación, la interacción, la progresión y la respuesta de los minijuegos dentro del entorno Windows definido, junto con una compilación ejecutable del prototipo. Se estableció desde la planificación que estos resultados corresponden a comprobación técnica y que no incluyen evaluación pedagógica, medición de aprendizaje ni validación con niños.

Como resultado documental se esperaba, además, un manual del usuario y un manual técnico de implementación que permitieran a un tercero utilizar el prototipo y mantener o ampliar el proyecto. La Tabla 4 relaciona cada objetivo específico con el resultado esperado y con la evidencia que lo respalda dentro del documento.

Tabla 4

Relación entre objetivos específicos, resultados esperados y evidencia documentada.

Objetivo específico	Resultado esperado	Evidencia en el documento
Crear los recursos gráficos 2D del videojuego.	Personajes, escenarios, objetos e interfaz organizados por mundo.	Apartado 4.2 y sus ilustraciones.
Diseñar la estructura de escenas, mundos y niveles del videojuego en Unity 2D.	Estructura de escenas y navegación funcional entre menú, selección, actividades y retorno.	Apartados 3.3 y 4.1.10, y tabla de escenas y flujo.
Programar en Unity 2D las mecánicas de interacción y la lógica de los minijuegos.	Mecánicas de clic y selección directa con validación y retroalimentación.	Apartados 4.1.6 y 4.1.7, y tabla de mecánicas implementadas.
Implementar mediante programación los sistemas complementarios del prototipo.	Control de avance, estrellas, activación visual de planetas, persistencia y reconocimiento de voz.	Apartados 3.4, 4.1.9 y tabla de datos y persistencia.
Evaluar el funcionamiento técnico mediante pruebas funcionales.	Registro de pruebas y compilación ejecutable para Windows.	Apartados 4.1.11 y 4.1.13, y tablas de resultados técnicos.

Nota. Elaboración propia.

1.12 Relación con el protocolo de investigación

El trabajo mantiene relación directa con el protocolo aprobado. Se conservan el problema, el objetivo general, los objetivos específicos y el alcance. La tesis presenta

Mundo Aprendo como videojuego educativo 2D para niños de 4 a 5 años, desarrollado en Unity y organizado mediante un flujo de trabajo basado en Kanban.

Tabla 5

Trazabilidad entre el protocolo aprobado y el trabajo de titulación.

Elemento del protocolo	Continuidad en la tesis
Título	Se presenta Mundo Aprendo como videojuego educativo 2D para niños de 4 a 5 años, desarrollado en Unity.
Problema	Se explica la necesidad de integrar actividades lúdicas digitales en un juego 2D organizado y comprensible.
Objetivo general	Se conserva la creación de un juego educativo funcional en Unity con recursos visuales y minijuegos.
Objetivos específicos	Se describen los recursos gráficos, las actividades, los mundos, el avance y las pruebas.
Metodología	Se organiza el desarrollo mediante un flujo de trabajo basado en Kanban, con un flujo continuo de tareas adecuado a un trabajo realizado por una sola persona.
Alcance	Se mantiene el uso en PC Windows; no se incluye venta, versiones web o móviles ni servicios en línea.

Nota. Elaboración propia con base en el protocolo aprobado.

Esta relación permite que los capítulos siguientes expliquen cómo se usa y cómo se desarrolla el juego, manteniendo la idea inicial del anteproyecto.

La correspondencia con el protocolo no se limita a los elementos formales. El proyecto Mundo Aprendo se originó dentro de un trabajo conjunto con el área de Educación, y esa procedencia condiciona la forma en que deben leerse los aportes documentados en esta tesis.

La problemática educativa que dio origen al videojuego fue planteada desde el área de Educación, al igual que la definición inicial de los personajes principales y de la paleta cromática. Determinadas decisiones se tomaron de manera conjunta, principalmente algunas mecánicas de las actividades, las reglas de puntuación, el sistema de estrellas y los criterios de avance vinculados con el contenido de cada mundo.

El aporte documentado en este trabajo de titulación corresponde al desarrollo tecnológico y artístico del prototipo. Comprende el desarrollo y la adaptación de los

recursos gráficos 2D, la programación en Unity y C#, la implementación de las mecánicas, la construcción de las interfaces, la navegación entre escenas, la integración de animaciones, efectos visuales y audio, los sistemas de progreso y puntuación, la persistencia local, la integración del reconocimiento de voz, la organización de escenas, scripts y componentes, las pruebas funcionales y la compilación final para Windows.

Durante el desarrollo se ajustaron algunos elementos respecto de la formulación inicial. La estructura de mundos y actividades se consolidó en cuatro temáticas, la interacción se concentró en el uso del mouse y el módulo de voz se resolvió mediante el servicio de dictado disponible en Windows. Estos ajustes se registran en el Capítulo IV junto con sus implicaciones técnicas.

Quedaron fuera del alcance de esta tesis los componentes del protocolo relacionados con la aplicación del recurso en aula, la evaluación pedagógica y cualquier medición sobre los niños. En consecuencia, el presente documento no reporta resultados de aprendizaje, atención, memoria ni desarrollo cognitivo, y limita sus afirmaciones al funcionamiento técnico comprobado del prototipo.

Capítulo II

2. Marco Teórico

Este capítulo reúne las fuentes y los fundamentos que sustentan las decisiones tomadas durante el desarrollo de Mundo Aprendo. Primero se analizan los antecedentes de investigación relacionados con videojuegos educativos, juego digital temprano y reconocimiento de habla infantil. Después se exponen las bases teóricas, metodológicas y tecnológicas del proyecto, las arquitecturas y estándares considerados, las herramientas empleadas y los conceptos técnicos utilizados. El capítulo cierra relacionando de forma explícita cada fundamento con una decisión implementada en el prototipo.

2.1 Antecedentes de investigación

Una revisión sistemática sobre aprendizaje basado en juegos identifica retos, recompensas, interactividad y retroalimentación inmediata entre los elementos que pueden sostener la participación; también advierte la necesidad de un diseño apropiado para la edad [4]. En el prototipo, estos criterios se traducen en instrucciones cortas, controles visibles y respuestas rápidas.

La revisión del juego digital en los primeros años incluye aplicaciones, tabletas, juegos y otros recursos, y muestra que la interacción depende del contexto en que se usa el dispositivo [5]. Un estudio de diseño para preescolares aplicó además un proceso iterativo en el que cada etapa recoge información y orienta la siguiente [7]. Estos aportes respaldan la construcción modular de Mundo Aprendo.

Los estudios revisados utilizan entradas distintas: un exergame convierte el movimiento corporal en datos del juego, mientras una revisión de juegos serios registra PC,

tabletas, teléfonos, cámaras, sensores y realidad virtual [3], [6]. Esta comparación permite justificar una arquitectura más limitada en Mundo Aprendo, basada en mouse y voz.

La implementación para Inicial II no se limitó a enumerar aplicaciones: seleccionó recursos a partir de sus características pedagógicas, usabilidad, seguridad y calidad general [11]. Este criterio es útil para Mundo Aprendo porque obliga a documentar las pantallas, las condiciones de uso y las funciones antes de integrar cada módulo.

Una revisión sobre tecnología en preescolar plantea un uso equilibrado y supervisado de las TIC, mientras un trabajo de educación inicial destaca la necesidad de contenido interactivo y comunicación con los educadores [12], [13]. Por ello, el juego utiliza actividades delimitadas y deja la configuración inicial bajo acompañamiento adulto.

El reconocimiento de habla infantil requiere recursos lingüísticos y acústicos ajustados a las voces de los niños. Un proyecto recopiló y anotó un corpus de habla infantil y trabajó con gramáticas, un léxico de pronunciación y modelos acústicos específicos [14]. En Mundo Aprendo, esta complejidad se reduce mediante textos esperados y un conjunto delimitado de palabras.

Otro estudio probó reconocimiento de voz en aulas preescolares naturales, donde aparecen conversaciones, música y otros ruidos [15]. Este antecedente justifica que la interfaz muestre el estado de escucha y ofrezca un reintento cuando la transcripción no coincide.

TurtleTalk convierte expresiones de los niños en acciones visibles en pantalla mediante una interfaz de voz [16]. Mundo Aprendo adopta esa relación entre entrada y resultado, pero la limita a validar la lectura del texto presentado, sin generar código ni utilizar una red neuronal propia.

Un estudio comparó un juego digital y una aplicación sin elementos de juego para practicar sonidos y palabras; el diseño mantuvo aspectos comunes y varió elementos como recompensas, retroalimentación y autonomía [17]. Esta diferencia permite describir las estrellas y los mensajes de Mundo Aprendo como componentes del sistema de juego, no como sustitutos de la actividad.

Una revisión distingue la exposición pasiva del uso activo e interactivo de pantalla y advierte que el tipo de contenido modifica la interpretación de los resultados [18]. En el prototipo, el clic, la selección y la voz se documentan como entradas directas porque activan reglas y generan cambios observables.

Un estudio midió tiempo de pantalla y funciones ejecutivas en 1016 preescolares mediante pruebas y cuestionarios reportados por sus madres [19]. Otro analizó 42 niños con una tarea Go/No-Go y un modelo de redes [20]. Ninguno de esos procedimientos forma parte de la validación de Mundo Aprendo; por ello, la tesis no atribuye cambios cognitivos al videojuego.

Un estudio aplicó cuestionarios a 431 padres y analizó la relación entre uso de videojuegos, función ejecutiva, desarrollo social y edad mediante un modelo estadístico [21]. Ese diseño exige participantes y análisis inferencial, mientras Mundo Aprendo solo realiza pruebas técnicas internas.

Una revisión sistemática basada en PRISMA seleccionó 16 estudios para examinar distintas funciones ejecutivas [22]. Sus resultados sirven como contexto documental y no como medición del prototipo.

Otros trabajos examinan neurorrehabilitación con robótica, realidad virtual y neuromodulación [23]. También describen técnicas como fMRI, PET, EEG, MEG y NIRS

para estudiar procesos neurales vinculados al aprendizaje [24]. Estas arquitecturas, dispositivos y métodos quedan fuera del alcance de una aplicación 2D local.

Los antecedentes revisados se sintetizan en la Tabla 6, que relaciona cada aporte principal con su correspondencia y sus límites respecto de Mundo Aprendo.

Tabla 6

Matriz de antecedentes y aportes al proyecto.

Ref.	Aporte principal	Relación con Mundo Aprendo	Diferencia o límite
Diseño e interacción	Interactividad, facilidad de uso, control visible y retroalimentación.	Orienta la forma de presentar acciones y respuestas en la interfaz.	Las fuentes usan juegos y contextos distintos.
Dispositivos y plataformas	Movimiento corporal, PC, tabletas, teléfonos, cámaras, sensores y realidad virtual.	Permite comparar esas entradas con el mouse y la voz del prototipo.	Mundo Aprendo no utiliza sensores externos ni realidad virtual.
Unity 2D y ciclo de desarrollo	Juego 2D construido en Unity mediante fases de desarrollo.	Respalda el motor y la organización técnica por etapas.	La temática y la mecánica del antecedente son diferentes.
Consignas y pasos	Actividades separadas en instrucciones, recursos, movimientos y secuencias.	Ayuda a ordenar la consigna, la entrada y la respuesta del minijuego.	La conversión a mecánicas digitales es propia del prototipo.
Contexto y aplicaciones digitales	Primera infancia, contexto digital y organización de aplicaciones.	Ubica el público y las condiciones generales de uso.	No define la arquitectura ni las mecánicas exactas del juego.
Reconocimiento de voz	Vocabulario delimitado, ambientes con ruido y comandos con respuesta visual.	Orienta el flujo técnico de la actividad de lectura.	Los idiomas y las tecnologías de las fuentes son diferentes.
Uso de pantalla y métodos de estudio	Comparación de aplicaciones, interacción activa y formas de registrar el uso.	Ayuda a describir las entradas directas sin ampliar el alcance del software.	Los métodos revisados no forman parte de las pruebas funcionales.
Enfoques tecnológicos distintos	Robótica, realidad virtual, análisis neural y generación automática de arte.	Sirven como contraste con una arquitectura 2D local y recursos predefinidos.	Esas tecnologías no se implementan en Mundo Aprendo.

Nota. Elaboración propia con base en [1], [25].

Los antecedentes recientes muestran soluciones que combinan videojuegos, aplicaciones móviles, sensores, reconocimiento de voz y entornos inmersivos. Sin embargo, la infraestructura y los métodos de evaluación varían de manera considerable. Para este

proyecto se priorizó una aplicación 2D local con mouse y un módulo de voz acotado, decisión que reduce dependencias y facilita la verificación de cada actividad por separado.

En conjunto, el estado del arte respalda el uso de instrucciones breves, retroalimentación inmediata, progresión visible y organización modular. También evidencia que la adecuación al público infantil debe comprobarse mediante evaluación específica. Por esta razón, los criterios revisados orientan el diseño de la propuesta, pero no se interpretan como prueba de aprendizaje ni de mejora cognitiva.

2.2 Bases teóricas

2.2.1 Videojuego educativo y aprendizaje basado en juegos

Los estudios revisados coinciden en que un juego educativo necesita metas claras, reglas comprensibles, control visible y retroalimentación inmediata [4], [7]. En Mundo Aprendo, estos criterios se convierten en una acción principal por pantalla, indicadores de estado y una respuesta audiovisual al terminar cada intento.

Los mundos y niveles organizan cuatro actividades: secuencias musicales, lectura apoyada por voz, comparación de tamaños y reconocimiento de emociones. Cada actividad tiene una consigna, una entrada, una validación y una salida.

2.2.2 Organización de las actividades del videojuego

La organización de los minijuegos es una decisión funcional del prototipo. Cada módulo separa la presentación de la consigna, la acción del usuario, la revisión de la respuesta y el cierre de la actividad.

Las actividades usan elementos visuales, textos cortos y sonidos. Esta estructura permite que la navegación, la validación y el progreso se programen de forma similar en los cuatro mundos.

2.2.3 Estructuración de actividades y consignas

Las fuentes sobre gimnasia cerebral presentan las actividades mediante instrucciones, recursos y una secuencia de ejecución. Una guía para educación inicial organiza ejercicios, y otro estudio aplica movimientos corporales con niños de 4 a 5 años [9], [10]. En el videojuego, esa forma ordenada se adapta a una consigna, una entrada y una validación.

Otra propuesta estructura el trabajo en planificación, justificación y control-monitoreo, mientras un estudio adicional describe instrucciones grupales y ejercicios corporales [26], [27]. Mundo Aprendo no reproduce esos ejercicios ni asume sus efectos; utiliza únicamente el principio de presentar pasos breves y visibles.

2.2.4 Interacción activa y respuesta del sistema

La revisión del juego digital en primera infancia muestra que la participación depende de las acciones del niño, del recurso y del contexto de uso [5]. En Mundo Aprendo, cada clic, selección o entrada de voz se recibe como un dato y se compara con una condición programada.

Las fuentes sobre tiempo de pantalla diferencian la interacción activa de la exposición pasiva y utilizan pruebas específicas para estudiar su relación con la atención y las funciones ejecutivas [18], [19]. La distinción permite describir el comportamiento técnico sin atribuir un beneficio no medido.

Un análisis de redes combina tiempo de pantalla, función ejecutiva y habilidades motoras en una muestra preescolar [20]. El prototipo no reproduce ese diseño: su verificación comprueba si la entrada cambia objetos, mensajes, sonidos o el estado de progreso.

El diseño iterativo para preescolares utiliza objetivos visibles, niveles progresivos, control del jugador y retroalimentación inmediata [7]. Por ello, cada acción del prototipo produce un cambio visual, un sonido o un mensaje breve que informa el resultado.

2.3 Bases metodológicas

2.3.1 Game Development Life Cycle (GDLC)

GDLC organiza la creación de videojuegos en fases como iniciación, preproducción, producción, pruebas, beta y lanzamiento [8]. Su especialización permite ordenar hitos artísticos y técnicos; sin embargo, una aplicación rígida resulta menos conveniente cuando es necesario regresar con frecuencia entre recursos gráficos, escenas, programación, integración y comprobación.

2.3.2 Scrum

Scrum estructura el trabajo en Sprints de duración fija y define responsabilidades, eventos y artefactos para inspeccionar y adaptar cada incremento [28]. Ofrece una cadencia regular, pero su estructura formal de equipo no correspondía plenamente a un desarrollo unipersonal con tareas heterogéneas y duraciones variables.

2.3.3 Kanban

Kanban busca optimizar el flujo mediante su definición y visualización, la gestión activa de los elementos de trabajo y la mejora del flujo [29]. Su continuidad facilita incorporar cambios y priorizar dependencias sin esperar el cierre de una iteración temporal. No obstante, una aplicación completa requiere controlar el trabajo en curso y medir el flujo.

2.3.4 Comparación y selección metodológica

Tabla 7

Comparación de metodologías de desarrollo consideradas.

Criterio	GDLC	Scrum	Flujo basado en Kanban
Organización	Etapas propias del desarrollo de videojuegos.	Sprints de duración fija con eventos definidos.	Flujo continuo y visible de tareas.
Roles	Dependen de la organización del proyecto.	Product Owner, Scrum Master y Developers.	No exige roles adicionales; apto para trabajo unipersonal.
Gestión de cambios	Los cambios pueden obligar a volver a fases previas.	Se priorizan en el Product Backlog para Sprints posteriores.	Se incorporan y priorizan según capacidad y dependencias.
Verificación	Pruebas concentradas en fases específicas, aunque admite iteración.	Inspección del incremento en cada Sprint.	Criterio de cierre verificable para cada tarea.
Adecuación al proyecto	Útil como referencia de fases generales del videojuego.	Aporta cadencia, pero su estructura formal resulta amplia para una sola persona.	Se ajusta a tareas artísticas y técnicas de duración variable.

Nota. Elaboración propia con base en [8], [28], [29].

A partir de la comparación, se seleccionó una adaptación operativa basada en Kanban porque permite visualizar y priorizar un flujo continuo sin exigir roles adicionales ni iteraciones de duración fija. Esta característica se ajustó al desarrollo unipersonal de Mundo Aprendo y a la coexistencia de actividades artísticas, de programación, integración y comprobación. GDLC se conservó como referencia de fases generales y Scrum se descartó como estructura principal. La selección responde a las condiciones del proyecto y no implica una superioridad universal.

La adaptación tuvo una limitación: no se conservaron límites históricos de trabajo en curso ni métricas de flujo, tiempo de ciclo, rendimiento o antigüedad de las tareas. En consecuencia, no se afirma el cumplimiento íntegro de Kanban ni se evalúa la productividad del proceso; se documenta únicamente el flujo operativo empleado para incorporar, priorizar y cerrar tareas verificables.

2.4 Bases tecnológicas

2.4.1 Motores de desarrollo de videojuegos

La selección tecnológica se examinó según el soporte para escenas y recursos 2D, el modelo de programación, la construcción de interfaz y audio, el destino Windows y el costo

de migración del código y de las pruebas. La comparación determina la correspondencia con Mundo Aprendo y no una superioridad general entre motores.

2.4.1.1 Unity

Unity ofrece un flujo 2D basado en escenas, GameObjects, componentes, sprites, interfaz, audio y scripts [30], [31] Estas características coinciden con la implementación verificada: seis escenas, 61 archivos C#, interfaz UGUI/TextMesh Pro, persistencia local, pruebas EditMode y PlayMode y una compilación para Windows.

2.4.1.2 Godot

Godot incorpora un renderizador y un motor físico dedicados a 2D, además de nodos, escenas, animación y herramientas de mapas; admite GDScript y C# [32], [33]. Para Mundo Aprendo, su adopción habría requerido reconstruir las escenas, trasladar la lógica existente y sustituir o adaptar la integración de voz dependiente de Unity y Windows.

2.4.1.3 Unreal Engine

Unreal Engine dispone de Paper 2D para sprites, Flipbooks y mapas de mosaicos, y permite desarrollar mediante Blueprint, C++ o una combinación de ambos [34], [35]. Su empleo habría exigido reimplementar la lógica C#, las escenas, los componentes de interfaz, la persistencia y las pruebas definidas para Unity.

2.4.2 Comparación y selección del motor

Tabla 8

Comparación de motores de desarrollo considerados.

Criterio	Unity	Godot	Unreal Engine
Soporte 2D	Flujo con escenas, GameObjects, componentes, sprites, UI y audio.	Renderizador y física 2D dedicados; nodos y escenas.	Paper 2D para sprites, Flipbooks y mapas de mosaicos.
Programación	C# integrado al modelo de componentes.	GDScript o C#, entre otras opciones.	Blueprint, C++ o combinación de ambos.
Ventaja para el caso	Coincide con los scripts, escenas, paquetes y pruebas existentes.	Flujo 2D especializado y flexible.	Amplio ecosistema y combinación de contenido 2D y 3D.
Costo para Mundo Aprendo	No requiere migración de la implementación verificada.	Exigiría reconstruir escenas, adaptar voz y trasladar la lógica.	Exigiría reimplementar C#, UI, persistencia y pruebas.

Criterio	Unity	Godot	Unreal Engine
Adecuación	Alta para la versión Windows documentada.	Viable para un desarrollo nuevo, no para migración sin costo.	Viable, pero sobredimensionado para el alcance 2D local actual.

Nota. Elaboración propia con base en [30], [35].

La comparación muestra que los tres motores permiten desarrollar contenido 2D. Unity fue la alternativa más adecuada para el alcance concreto por su correspondencia con C#, el modelo de escenas y componentes, la compilación Windows y el Test Framework ya presentes en el proyecto. Mantenerlo evita una migración que no añade una función exigida por el alcance. Esta selección responde a compatibilidad y continuidad técnica, no a una superioridad universal.

2.4.3 Lenguajes y entornos de programación

Unity emplea C#; Godot admite GDScript y C#; y Unreal Engine combina Blueprint con C++ [31], [33],. Para Mundo Aprendo se seleccionó C# porque constituye la base comprobada de los 61 scripts y de su integración con los componentes de Unity. Además del lenguaje, se compararon los entornos de edición que Unity 6 reconoce para el trabajo con código C#: Visual Studio, Visual Studio Code y JetBrains Rider [36].

2.4.3.1 Visual Studio

En Windows, Visual Studio es el entorno predeterminado para Unity. Visual Studio Tools for Unity incorpora depuración especializada, IntelliSense, refactorizaciones, acceso contextual a la documentación y referencias de uso mediante CodeLens [36], [37]. Su integración directa con Unity y C# reduce la configuración inicial en el sistema operativo utilizado por el proyecto.

2.4.3.2 Visual Studio Code

Visual Studio Code es compatible con Unity en Windows, macOS y Linux. Para disponer de análisis y depuración requiere las extensiones de C#, C# Dev Kit y Unity, además del paquete Visual Studio Editor de Unity; el antiguo paquete com.unity.ide.vscode

ya no cuenta con soporte [36]. Constituye una alternativa modular, aunque su integración completa depende de esa configuración adicional.

2.4.3.3JetBrains Rider

JetBrains Rider es un entorno multiplataforma de pago con soporte específico para Unity, que incluye inspecciones, acciones rápidas, autocompletado, depuración de scripts y ejecución de pruebas [36], [38]. Resulta pertinente cuando se prioriza el análisis especializado del código y se dispone de la licencia correspondiente.

2.4.3.4Comparación y selección del entorno

Para la continuidad técnica de Mundo Aprendo se selecciona Visual Studio como entorno de referencia, debido a que el proyecto se mantiene en Windows, utiliza Unity y C#, y esa combinación dispone de integración oficial para edición y depuración [36], [37]. Visual Studio Code y Rider permanecen como alternativas compatibles. Esta selección orienta el mantenimiento posterior y no se presenta como evidencia histórica del editor usado durante toda la construcción, porque la versión examinada no conserva un registro concluyente de esa preferencia.

2.4.4 Mecánicas de interacción

Las actividades utilizan clic, selección de opciones, repetición ordenada de entradas y respuesta a instrucciones breves. Cada minijuego dispone de un inicio, una acción esperada, una regla de validación y retroalimentación visual o sonora. Esta organización permite comprobar entradas y salidas observables sin interpretar los resultados como mediciones de usabilidad infantil.

2.4.5 Reconocimiento de voz y flujo de interacción

El módulo de voz opera con textos esperados y un vocabulario delimitado. Los recursos lingüísticos y acústicos específicos para habla infantil muestran por qué no conviene aceptar una entrada abierta en esta versión [14]. El ruido de aula, las conversaciones y otros sonidos pueden afectar el reconocimiento [15]; por ello, la interfaz informa el estado de escucha, muestra la transcripción y permite repetir.

La secuencia técnica comprende la indicación, la activación del micrófono, la lectura, la presentación del texto reconocido, su normalización, la comparación con el texto esperado y la respuesta audiovisual. La comprobación documentada se limita a ese flujo y a las condiciones del equipo; no establece precisión general del reconocimiento con niños.

2.5 Arquitecturas, modelos y estándares

La arquitectura se estudió mediante alternativas aplicables al tipo de solución y se contrastó con la implementación real. El objetivo fue describir responsabilidades y dependencias verificables, sin atribuir al código un patrón formal que no se encuentre representado en sus escenas, clases y componentes.

2.5.1 Alternativas arquitectónicas

2.5.1.1 Arquitectura en capas

La arquitectura en capas separa responsabilidades y orienta dependencias entre presentación, lógica, servicios y datos [39]. En Mundo Aprendo permite interpretar las escenas y vistas como presentación, los controladores de actividad como lógica, la navegación, el audio y la voz como servicios, y los repositorios con PlayerPrefs como persistencia.

2.5.1.2 Modelo–Vista–Controlador (MVC)

MVC divide la interacción en modelo, vista y controlador para separar asuntos y facilitar su comprobación [40]. Aunque algunos elementos del prototipo pueden interpretarse con esas funciones, no se seleccionó como descripción formal porque varios componentes de Unity combinan eventos, presentación y comportamiento de escena; declarar un MVC estricto no representaría fielmente la implementación.

2.5.1.3 Arquitectura basada en componentes y eventos

La organización por componentes coincide con el modelo de Unity, en el que los scripts se incorporan a GameObjects para proporcionar comportamiento [30], [31]. Los eventos permiten que controladores y elementos visuales intercambien estados sin concentrar toda la lógica en una sola clase. Su límite es que el modelo de componentes, por sí solo, no establece la dirección de todas las dependencias.

2.5.2 Comparación y selección arquitectónica

Tabla 9

Comparación de alternativas arquitectónicas aplicables.

Criterio	Arquitectura en capas	MVC	Componentes y eventos
Separación	Agrupa presentación, lógica, servicios y persistencia.	Divide modelo, vista y controlador.	Compone comportamientos en GameObjects y comunica cambios por eventos.
Ventaja	Orienta dependencias y responsabilidades.	Favorece separación conceptual y comprobación de responsabilidades.	Coincide con el modelo nativo de Unity y facilita reutilización.
Límite	Puede introducir estructura innecesaria en módulos pequeños.	No representa con precisión componentes de escena que combinan presentación y comportamiento.	Por sí sola no define la dirección de las dependencias.
Aplicación al proyecto	Capas lógicas para presentación, actividades, servicios y persistencia.	Se usa solo como alternativa comparada; no se declara MVC estricto.	Base de implementación mediante MonoBehaviour, servicios y eventos.

Nota. Elaboración propia con base en [30], [40].

Como resultado se seleccionó una arquitectura modular por capas lógicas, implementada sobre el modelo de componentes de Unity y complementada con servicios y eventos. Esta combinación representa la estructura real descrita en el Capítulo IV y permite

separar presentación, control de actividades y funciones transversales sin atribuir al proyecto un MVC completo.

2.6 Herramientas de desarrollo

Las herramientas utilizadas se derivan de la comparación tecnológica y de la arquitectura seleccionada. Unity y C# soportan las escenas, los componentes y los servicios; los paquetes de renderizado, interfaz y entrada resuelven la presentación; el Test Framework aporta comprobaciones técnicas; Windows define el destino de compilación y condiciona la voz; y las herramientas gráficas preparan los recursos antes de importarlos.

Tabla 10

Herramientas y recursos tecnológicos utilizados.

Herramienta o recurso	Uso verificado dentro del proyecto	Justificación de la selección
Unity Editor 6000.4.0f1	Escenas, GameObjects, componentes, recursos, interfaz y compilación del videojuego 2D.	Coincide con la implementación, el destino Windows y las pruebas ya disponibles.
C# y scripting de Unity	61 scripts para controladores, servicios, repositorios, mecánicas y presentación.	Se integra con el modelo de componentes verificado y evita migrar la lógica existente.
URP 17.4.0	Canal de renderización configurado para la presentación 2D.	Corresponde a la configuración comprobada del proyecto.
UGUI / TextMesh Pro 2.0.0 e Input System 1.19.0	Interfaz, textos y entrada del usuario.	Resuelven los controles y la presentación documentados en las escenas.
Unity Test Framework 1.6.0	Ejecución de casos EditMode y PlayMode.	Permite comprobar componentes y escenas dentro del mismo entorno.
Windows Standalone x86-64 con Mono	Plataforma y backend de la compilación de escritorio.	Constituyen el destino técnico delimitado para el prototipo.
DictationRecognizer y UnityEngine.Microphone	Captura, transcripción y validación acotada de la actividad de lectura.	Corresponden a la integración de voz verificada y a su dependencia de Windows.
Adobe Illustrator, Adobe Photoshop y PNG	Preparación de fuentes gráficas y exportación de sprites con transparencia.	Permiten conservar editables y entregar recursos compatibles con Unity.

Nota. Elaboración propia a partir de la comparación tecnológica y de la configuración verificada del proyecto.

A partir de la comparación de motores, se seleccionó Unity por su correspondencia con la implementación 2D, C#, el modelo de componentes, el destino Windows y la campaña de pruebas disponible. A partir de la comparación de lenguajes, se mantuvo C# porque constituye la base de la lógica comprobada. Las versiones y configuraciones de la tabla coinciden con las verificadas en el Capítulo IV.

La Tabla 10 conserva únicamente las herramientas cuya presencia pudo verificarse en el proyecto. Visual Studio se seleccionó en las bases tecnológicas como entorno de

referencia para el mantenimiento en Windows [36], [37], pero no se registra como herramienta histórica de construcción porque no se encontró evidencia concluyente del editor externo utilizado. Migrar a Godot o Unreal habría requerido trasladar 61 scripts, reconstruir seis escenas, adaptar la voz y repetir las pruebas y la compilación sin añadir una función exigida por el alcance.

2.7 Marco conceptual

Los siguientes conceptos se presentan como definiciones operativas del proyecto y facilitan la lectura de los capítulos que describen el uso y la construcción del juego. La Tabla 11 reúne los términos pedagógicos, metodológicos, de diseño de videojuegos y de desarrollo técnico empleados en el documento.

Tabla 11

Glosario de términos y conceptos utilizados en el documento.

Término o concepto	Definición operativa utilizada en el documento
Animator	Componente y sistema de Unity utilizado para organizar estados, clips y transiciones de animación.
Aprendizaje basado en juegos	Enfoque que incorpora reglas, retos, objetivos y retroalimentación propios del juego dentro de una actividad con finalidad educativa.
Arquitectura modular	Organización del programa en partes con responsabilidades diferenciadas para facilitar integración, prueba y mantenimiento.
Backend de scripting	Tecnología que Unity utiliza para convertir y ejecutar el código C# en la compilación; en el proyecto corresponde a la configuración de la plataforma Windows.
C#	Lenguaje de programación empleado para implementar la lógica, los controladores y los sistemas del videojuego en Unity.
Canal alfa	Información de transparencia incluida en una imagen, utilizada para recortar visualmente sprites y elementos de interfaz.
Canvas	Contenedor de Unity en el que se organizan los elementos de la interfaz de usuario, como botones, textos, paneles e indicadores.
Compilación	Proceso mediante el cual Unity genera el archivo ejecutable y los datos necesarios para utilizar el videojuego en Windows.
Consigna	Instrucción breve que indica al usuario la acción que debe realizar dentro de una actividad.
Entrada de usuario	Dato recibido por el sistema a partir del mouse, el micrófono u otro dispositivo de interacción.

Término o concepto	Definición operativa utilizada en el documento
Escena	Archivo de Unity que reúne objetos, componentes y recursos correspondientes a una pantalla, menú o actividad.
Exergame	Videojuego que integra actividad física o movimiento corporal como parte de su interacción; se utiliza en la tesis como antecedente comparativo y no como característica implementada.
FPS	Cantidad de fotogramas mostrados por segundo durante la ejecución de una aplicación; se usa como indicador técnico de fluidez.
Funciones ejecutivas	Conjunto de procesos como atención, memoria de trabajo, inhibición y flexibilidad cognitiva mencionados en las fuentes; no fueron medidos en esta tesis.
GameObject	Objeto base de una escena de Unity al que se asignan componentes, scripts, imágenes, audio u otras funciones.
Gimnasia cerebral	Referencia temática proveniente del área de Educación para organizar actividades de estimulación; esta tesis documenta el desarrollo técnico y no comprueba sus efectos pedagógicos.
Inspector	Panel de Unity que permite revisar y configurar los componentes, valores y referencias asignados a un GameObject.
Interacción digital	Acción realizada mediante un dispositivo de entrada que produce una respuesta observable en la aplicación.
Interfaz de usuario (UI)	Conjunto de pantallas, botones, textos, indicadores y elementos visuales mediante los cuales el usuario se comunica con el videojuego.
Jerarquía	Panel y estructura de Unity que muestra los GameObjects contenidos en una escena y la relación de dependencia entre ellos.
Juego serio	Videojuego diseñado con una finalidad principal distinta del entretenimiento, como educación, entrenamiento o apoyo terapéutico.
Minijuego	Módulo breve con una consigna, una entrada, una regla de validación y un resultado.
Mundo	Unidad temática del videojuego que agrupa una actividad, su ambientación, sus recursos y su sistema de progreso.
Neuroplasticidad	Capacidad del sistema nervioso para modificar su organización y funcionamiento en respuesta a experiencias; se aborda desde las fuentes y no constituye una variable medida en el proyecto.
Nivel	Estado o etapa de avance dentro de una actividad; en el documento no se utiliza como sinónimo de mundo.
Persistencia local	Conservación de datos de avance en el equipo del usuario, sin depender de una base de datos o servidor externo.
Pipeline	Secuencia organizada de tareas y herramientas utilizada para producir, preparar, importar e integrar recursos dentro del proyecto.
PlayerPrefs	Sistema de Unity utilizado para guardar valores simples de configuración y progreso en el equipo local.
PNG	Formato de imagen que admite compresión sin pérdida y transparencia mediante canal alfa.
Prefab	Recurso reutilizable de Unity que almacena un GameObject configurado con sus componentes y elementos asociados.

Término o concepto	Definición operativa utilizada en el documento
Profiler	Herramienta de Unity para observar consumo de recursos, tiempos de procesamiento y comportamiento de la aplicación durante la ejecución.
Progresión	Registro y representación del avance del usuario entre actividades, estrellas y mundos.
Prototipo funcional	Versión ejecutable que integra las funciones principales necesarias para demostrar y comprobar técnicamente la propuesta.
Prueba funcional	Comprobación de una acción, una condición y la respuesta esperada del software.
Reconocimiento de voz	Proceso mediante el cual una entrada de audio se transforma en texto para compararlo con palabras esperadas.
Retroalimentación	Respuesta visual, sonora o escrita que informa el resultado de una acción realizada por el usuario.
Revisión sistemática	Método de investigación documental que identifica, selecciona y sintetiza estudios mediante criterios explícitos; aparece en los antecedentes consultados.
Script	Archivo de código C# que define comportamientos, reglas o servicios y se asigna a objetos del proyecto.
Sprite	Imagen 2D utilizada para representar personajes, fondos, botones, iconos u objetos.
TextMesh Pro	Sistema de Unity empleado para mostrar texto con mayor control tipográfico y de representación visual.
Tiempo de pantalla	Duración de exposición o uso de dispositivos con pantalla; se presenta como concepto de los estudios revisados y no fue medido en el proyecto.
UGUI	Sistema de interfaz de usuario de Unity basado en Canvas y componentes como Button, Image y Text.
Unity	Motor y editor utilizado para organizar escenas, recursos, interfaces, audio y programación del videojuego 2D.
URP	Universal Render Pipeline; canal de renderización de Unity configurado para procesar la presentación gráfica del proyecto.
Usabilidad	Grado en que una interfaz permite realizar tareas con claridad, eficiencia y comprensión dentro de un contexto de uso.
Videojuego educativo	Aplicación digital que combina reglas, actividades, interacción y retroalimentación con una finalidad educativa.
Windows Standalone	Plataforma de compilación de Unity utilizada para generar una aplicación de escritorio ejecutable en Windows.

La tabla concentra definiciones operativas para evitar que un mismo término se interprete de manera distinta entre los capítulos. En este documento, mundo designa la unidad temática; actividad o minijuego, la tarea interactiva asociada; escena, el contenedor técnico de Unity; y planeta, la representación visual de un mundo dentro del selector. El

término nivel se reserva para el estado de avance dentro de una actividad y no se emplea como sinónimo de mundo.

2.8 Relación entre la teoría y la propuesta tecnológica

Las ideas revisadas se reflejan en decisiones de diseño de Mundo Aprendo. Los mundos, las actividades, la interacción y los recursos se relacionan con el problema, los objetivos y el perfil del usuario. La Tabla 12 muestra esta correspondencia.

Tabla 12

Relación entre criterios de diseño y decisiones del videojuego.

Fundamento	Decisión en Mundo Aprendo	Resultado esperado
Interfaz clara	Usar controles visibles, una acción principal y respuestas breves.	Pantallas con navegación y retroalimentación comprensibles.
Organización modular	Separar menú, mundos, minijuegos, progreso, audio y voz.	Módulos que pueden revisarse por separado.
Consignas y pasos	Dividir cada actividad en instrucción, entrada, validación y cierre.	Flujo similar en los cuatro minijuegos.
Plataformas y entradas	Limitar el sistema a Windows, mouse y voz.	Sin sensores externos, realidad virtual ni servicios permanentes.
Contexto digital	Definir el grupo de edad y las condiciones básicas de uso.	Alcance centrado en un prototipo local.
Voz infantil	Usar palabras delimitadas, estado de escucha y respuesta visual.	Actividad de lectura con una alternativa para continuar.
Interacción y límites	Documentar solo las entradas, respuestas y tecnologías implementadas.	Pruebas funcionales sobre el software construido.

Nota. Elaboración propia con base en [1], [25].

En conclusión, las bases revisadas ayudan a organizar un videojuego educativo infantil con actividades lúdicas digitales. La metodología indica el orden de trabajo y Unity 2D permite convertir esas ideas en una versión funcional para PC Windows.

Capítulo III

3. Manual Del Usuario

Este capítulo documenta la experiencia de uso de Mundo Aprendo desde la perspectiva de quien lo utiliza. Describe los requisitos mínimos, el procedimiento de inicio, la navegación, los controles disponibles, las funciones de cada mundo y la retroalimentación que devuelve el sistema. No incluye clases, código ni detalles de arquitectura, los cuales se presentan en el Capítulo IV.

3.1 Presentación de la propuesta y perfil del usuario

Mundo Aprendo es un videojuego educativo 2D desarrollado en Unity para computadoras con sistema operativo Windows. La experiencia se organiza en cuatro mundos temáticos: el Mundo Musical, con actividades de secuencias musicales; el Mundo de los Cuentos, con lectura apoyada por reconocimiento de voz; el Mundo de los Tamaños, con comparación entre elementos grandes y pequeños; y el Mundo de las Emociones, con reconocimiento de expresiones emocionales.

El usuario directo es un niño de 4 a 5 años. Por esa razón, la interacción se resolvió mediante el mouse, con botones amplios, consignas breves y respuestas visibles después de cada acción. El videojuego no requiere que el niño lea de forma autónoma: las instrucciones se acompañan de imágenes, personajes y audio. Se prevé el acompañamiento de una persona adulta, docente, familiar o cuidadora, especialmente durante el primer uso y durante la actividad que utiliza el micrófono.

El personaje guía es Luli, quien aparece en el menú principal y en la pantalla de selección para presentar la experiencia y orientar al usuario. Cada mundo cuenta además con personajes propios que acompañan la actividad correspondiente.

Los requisitos mínimos comprobados para ejecutar el prototipo se resumen en la Tabla 13. Corresponden a la configuración en la que se verificó el funcionamiento y no a un rango de compatibilidad certificado.

Tabla 13

Requisitos mínimos comprobados para la ejecución de Mundo Aprendo.

Elemento	Condición comprobada
Sistema operativo	Windows de 64 bits.
Arquitectura de compilación	Windows Standalone x86-64.
Resolución base	1024 × 768, con comprobación visual en 1920 × 1080, 1366 × 768 y 1280 × 720.
Dispositivo de entrada principal	Mouse.
Dispositivo adicional	Micrófono predeterminado del sistema, requerido únicamente por el Mundo de los Cuentos.
Salida de audio	Altavoces o auriculares activos.
Instalación	No requiere instalador. Se ejecuta desde la carpeta de compilación completa.
Conexión a internet	No se requiere. El progreso se guarda de forma local.

Nota. Elaboración propia a partir de la configuración de compilación verificada.

3.2 Acceso e inicio de la experiencia

El acceso a Mundo Aprendo se realiza desde la carpeta de compilación generada para Windows. Esa carpeta debe conservarse completa, ya que el archivo ejecutable depende de los archivos de datos y de las bibliotecas que la acompañan. Si se copia únicamente el ejecutable, la aplicación no inicia.

Para iniciar la experiencia, la persona adulta abre la carpeta de compilación y ejecuta el archivo del videojuego. La aplicación arranca en pantalla completa y carga directamente el menú principal, sin pantallas de registro ni solicitud de datos personales.

Antes del primer uso conviene comprobar que la salida de audio funcione, ya que las consignas y la retroalimentación se apoyan en sonido. Si se va a utilizar el Mundo de los Cuentos, además debe verificarse que el micrófono predeterminado esté disponible y que los permisos de voz de Windows estén habilitados.

El menú principal presenta el título del videojuego, al personaje Luli en el centro de la pantalla y cuatro planetas que funcionan como botones: JUGAR, CONTROLES, OPCIONES y SALIR. La opción JUGAR conduce a la pantalla de selección de mundos; CONTROLES y OPCIONES muestran información de apoyo y ajustes; SALIR cierra la aplicación. La Ilustración 1 presenta esta pantalla.

Ilustración 1

Menú principal de Mundo Aprendo.

Nota. Captura propia obtenida de la versión comprobada del prototipo.

El menú concentra en una sola vista todas las acciones disponibles al inicio. La disposición de los botones alrededor del personaje y el uso de una etiqueta por planeta permiten que la persona adulta identifique la opción correcta sin recorrer submenús.

3.3 Navegación y controles

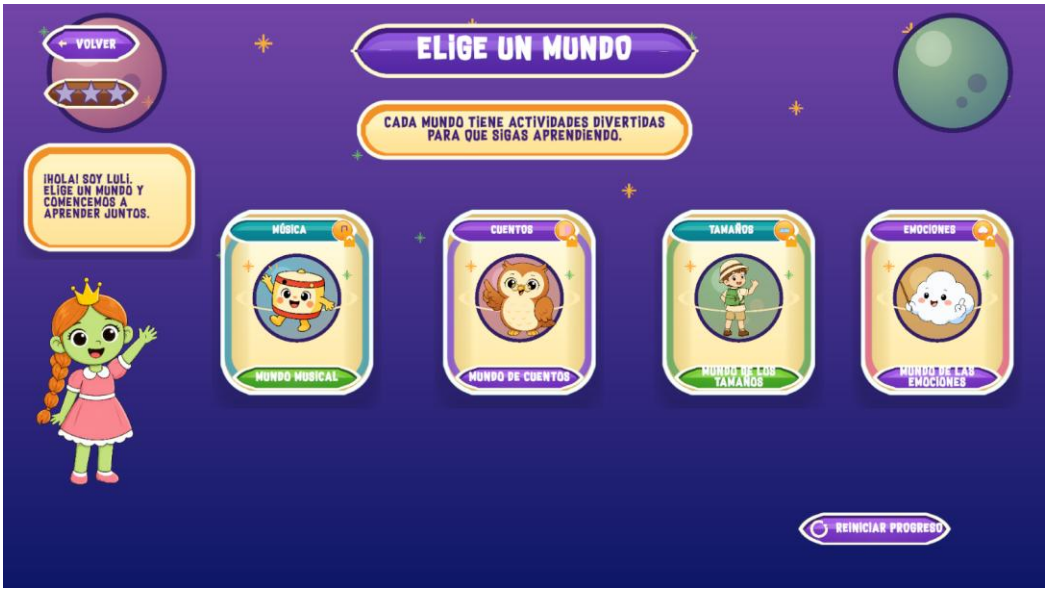
La navegación de Mundo Aprendo se realiza en su totalidad con el mouse. El usuario desplaza el cursor sobre el elemento deseado y confirma la acción con el clic izquierdo. No se requieren combinaciones de teclas, atajos ni dispositivos adicionales para

recorrer la experiencia; el micrófono interviene únicamente en la actividad de lectura del Mundo de los Cuentos.

Desde el menú principal, la opción JUGAR abre la pantalla de selección de mundos. Esta pantalla presenta el mensaje de bienvenida de Luli, un contador de estrellas acumuladas, las cuatro tarjetas correspondientes a los mundos temáticos, un botón VOLVER que retorna al menú y un botón REINICIAR PROGRESO que restablece el avance guardado. La Ilustración 2 muestra esta pantalla.

Ilustración 2

Pantalla de selección de mundos.



Nota. Captura propia obtenida de la versión comprobada del prototipo.

Cada tarjeta identifica el mundo mediante un color, un nombre y el personaje asociado. El botón de reinicio se ubicó en una zona alejada de las tarjetas para reducir la posibilidad de que el niño lo active de forma accidental.

Los mundos no se encuentran todos disponibles desde el inicio. Las tarjetas muestran un candado cuando el mundo permanece bloqueado y presentan su estado activo cuando el avance registrado permite ingresar. De este modo, el orden de las actividades se

comunica visualmente al usuario sin necesidad de texto adicional. La Ilustración 3 compara ambos estados.

Ilustración 3

Estados bloqueado y disponible de los mundos.

Nota. Captura propia obtenida de la versión comprobada del prototipo.

La diferencia entre el estado bloqueado y el disponible se apoya en el candado y en el tratamiento visual de la tarjeta, de manera que el usuario reconozca la condición del mundo antes de intentar abrirlo.

Dentro de cada mundo, la interfaz mantiene una disposición constante. En la parte superior se muestran el nombre del mundo y la consigna de la actividad; en una esquina se ubica el botón VOLVER, que retorna a la pantalla de selección; y en la esquina opuesta se presentan las estrellas obtenidas y el contador de errores de la ronda en curso. Esta constancia permite que el usuario reconozca los mismos elementos en las cuatro actividades. La Tabla 14 resume los controles disponibles.

Tabla 14

Controles e interacciones disponibles para el usuario.

Control	Acción del usuario	Resultado observado
Botones del menú principal	Clic izquierdo sobre el planeta correspondiente.	Abre la selección de mundos, muestra la información de controles y opciones, o cierra la aplicación.
Tarjetas de mundo	Clic izquierdo sobre la tarjeta disponible.	Abre el tutorial y la actividad del mundo seleccionado. Las tarjetas bloqueadas no responden.
Botón VOLVER	Clic izquierdo.	Retorna a la pantalla anterior sin perder el progreso guardado.
Botón SIGUIENTE del tutorial	Clic izquierdo.	Avanza a la siguiente instrucción del tutorial hasta habilitar el inicio de la actividad.
Botón INICIAR	Clic izquierdo.	Comienza la ronda de la actividad.
Elementos de la actividad	Clic izquierdo sobre el elemento u opción correspondiente.	Registra la respuesta del usuario y ejecuta la validación correspondiente.
Micrófono	Lectura en voz alta durante el Mundo de los Cuentos.	Captura la voz mediante el servicio de dictado de Windows y compara el texto reconocido con el esperado.
Botón REINICIAR PROGRESO	Clic izquierdo en la selección de mundos.	Restablece el avance y las estrellas registradas de forma local.

Nota. Elaboración propia a partir de la versión comprobada del prototipo.

3.4 Funciones e interacciones disponibles

Antes de comenzar cada actividad, el videojuego presenta un tutorial compuesto por instrucciones breves. Las instrucciones se muestran de forma secuencial en un panel central que indica el número de paso, y el usuario avanza mediante el botón SIGUIENTE. Al concluir el tutorial se habilita el botón INICIAR, que da paso a la actividad. La Ilustración 4 muestra este recurso en el Mundo Musical.

Ilustración 4

Tutorial e instrucciones previas a la actividad en Mundo Musical.

Nota. Captura propia obtenida de la versión comprobada del prototipo.

Ilustración 5

Tutorial e instrucciones previas a la actividad en Mundo de los Cuentos.

Nota. Captura propia obtenida de la versión comprobada del prototipo.

Ilustración 6

Tutorial e instrucciones previas a la actividad en Mundo de los Tamaños.

Nota. Captura propia obtenida de la versión comprobada del prototipo.

El tutorial cumple una función de acompañamiento: mantiene la actividad inactiva hasta que las instrucciones han sido presentadas, lo que permite a la persona adulta explicar la consigna antes de que el niño comience.

En el Mundo Musical, el sistema reproduce una secuencia y la representa mediante el encendido de los elementos correspondientes. A continuación, el usuario debe repetir esa secuencia haciendo clic en el mismo orden. El sistema compara la entrada con la secuencia esperada, informa el resultado y permite reintentar sin penalización de vidas. La Ilustración 7 presenta la actividad en ejecución.

Ilustración 7

Actividad de secuencias musicales en ejecución.

Nota. Captura propia obtenida de la versión comprobada del prototipo.

La actividad combina una consigna auditiva y una visual, de modo que el usuario disponga de dos referencias para reconocer el orden solicitado.

En el Mundo de los Cuentos, el usuario selecciona un cuento y lee en voz alta el texto que se presenta en pantalla. La interfaz indica cuándo el micrófono está escuchando y muestra el estado de la lectura. El sistema utiliza el servicio de dictado de Windows para transformar la voz en texto y lo compara con las palabras esperadas. La Ilustración 8 muestra la interfaz durante la escucha.

Ilustración 8

Actividad de lectura apoyada por reconocimiento de voz.

Nota. Captura propia obtenida de la versión comprobada del prototipo.

Esta actividad depende de las condiciones del entorno. El ruido ambiente, la distancia al micrófono y la configuración de idioma del sistema influyen en el reconocimiento, por lo que se recomienda utilizarla en un espacio silencioso y con acompañamiento adulto.

En el Mundo de los Tamaños, cada ronda presenta una consigna y dos opciones. El usuario debe seleccionar el elemento que corresponde a la indicación, por ejemplo el animal más grande. El sistema valida la selección, muestra la retroalimentación y avanza a la siguiente ronda. La Ilustración 9 presenta esta actividad.

Ilustración 9

Actividad de comparación de tamaños.

Nota. Captura propia obtenida de la versión comprobada del prototipo.

La consigna se mantiene visible durante toda la ronda y destaca la palabra clave de la comparación, de manera que el usuario pueda volver a ella si duda de la instrucción.

En el Mundo de las Emociones, el sistema muestra una expresión y el usuario selecciona el elemento que la representa entre las opciones disponibles. La actividad se desarrolla a lo largo de varias rondas y registra los aciertos y los errores acumulados. La Ilustración 10 muestra esta actividad.

Ilustración 10

Actividad de reconocimiento de emociones.

Nota. Captura propia obtenida de la versión comprobada del prototipo.

La actividad presenta una sola expresión por ronda, lo que concentra la atención del usuario en una única comparación y evita la superposición de estímulos.

Las cuatro actividades comparten el mismo esquema de funcionamiento: se presenta una consigna, el usuario realiza una acción, el sistema valida la respuesta y devuelve una retroalimentación visual y sonora. Cuando la respuesta es correcta, la interfaz lo confirma mediante un mensaje y un sonido; cuando no lo es, el sistema lo indica y permite intentar nuevamente. La Ilustración 11 presenta un ejemplo de retroalimentación positiva junto con la actualización del progreso.

Ilustración 11

Retroalimentación positiva al completar una actividad.

Nota. Captura propia obtenida de la versión comprobada del prototipo.

Ilustración 12

Actualización del progreso en la pantalla de selección de mundos.

Nota. Captura propia obtenida de la versión comprobada del prototipo.

La posibilidad de reintentar sin perder la partida es una decisión de diseño orientada a la edad del usuario: el error no interrumpe la actividad y solo interviene en el cálculo de las estrellas obtenidas.

3.5 Evidencia de uso

Este apartado presenta un recorrido documentado de uso, desde el inicio de la aplicación hasta el cierre de una actividad y la actualización del progreso. Las capturas corresponden a la versión comprobada del prototipo y muestran el funcionamiento tal como lo percibe el usuario final.

El recorrido comienza en el menú principal, continúa con la selección de un mundo disponible, avanza por el tutorial y la actividad, y concluye con la pantalla de resultado. Al finalizar una actividad, el sistema muestra un mensaje de cierre, informa la cantidad de estrellas obtenidas sobre un máximo de tres y ofrece dos opciones: repetir la actividad o regresar a la selección de mundos. La Ilustración 13 presenta esta pantalla.

Ilustración 13

Pantalla de cierre de una actividad con las estrellas obtenidas.

Nota. Captura propia obtenida de la versión comprobada del prototipo.

La pantalla de cierre reúne en un solo lugar el resultado y las dos continuaciones posibles, lo que evita que el usuario deba buscar la forma de salir de la actividad.

El progreso obtenido se refleja en la pantalla de selección de mundos. Las estrellas acumuladas se muestran en el contador superior y los mundos que cumplen la condición de avance pasan a su estado disponible. La Ilustración 14 documenta la actualización del progreso tras completar una actividad.

Ilustración 14

Progreso por estrellas en la pantalla de selección de mundos.

Nota. Captura propia obtenida de la versión comprobada del prototipo.

El avance se conserva de forma local entre sesiones, de modo que al volver a abrir el videojuego el usuario encuentra las estrellas obtenidas y los mundos ya desbloqueados. El botón REINICIAR PROGRESO permite volver al estado inicial cuando se desea comenzar de nuevo.

El recorrido documentado permite comprobar que la navegación, la ejecución de las actividades, la retroalimentación y el registro del avance funcionan de forma encadenada. Se trata de evidencia de funcionamiento técnico observable; no constituye una evaluación de usabilidad ni una medición de aprendizaje, las cuales no formaron parte del alcance de este trabajo.

Capítulo IV

4. Manual Técnico De Implementación

Este capítulo documenta el proceso técnico de construcción de Mundo Aprendo. Se organiza en dos bloques. El primero corresponde al manual de arte e integración visual y describe la dirección de arte, la producción de recursos 2D, la preparación de personajes, la adaptación de elementos de interfaz y la incorporación de los assets dentro de Unity. El segundo corresponde al manual del programador y reúne la arquitectura, las tecnologías, la organización del proyecto, el desarrollo de sistemas y mecánicas, la gestión de datos, la configuración de escenas, las pruebas, el despliegue y los procedimientos de ampliación.

Ambos bloques describen la versión comprobada del prototipo. Las afirmaciones se limitan a lo que puede verificarse en el proyecto de Unity, en el código fuente y en sus registros técnicos, de modo que el manual mantenga correspondencia entre la implementación documentada y el comportamiento observado durante la ejecución.

4.1 Manual de arte e integración visual

Este bloque documenta la producción visual del prototipo y su integración en el motor. Se presenta antes del manual del programador porque la producción de personajes, fondos, elementos de interfaz y variantes de pose constituyó una entrada directa para la construcción posterior de las escenas y de las mecánicas. La descripción diferencia las referencias conceptuales iniciales, los recursos reelaborados por el autor y los assets de terceros que fueron adaptados para mantener la identidad visual de Mundo Aprendo.

El proceso artístico no consistió en incorporar de forma directa las imágenes de referencia al videojuego. Parte de la conceptualización inicial fue proporcionada por la compañera del área de Educación mediante imágenes generadas con inteligencia artificial

que servían únicamente como guía aproximada de personajes, mundos y ambientaciones. A partir de estas referencias se reconstruyeron y adaptaron los recursos necesarios para la versión 2D del proyecto, evitando que la imagen de referencia se convirtiera en el archivo final utilizado en Unity.

Adobe Illustrator se empleó principalmente para reconstruir las formas, contornos y proporciones de los personajes mediante elementos vectoriales. Trabajar sobre un original vectorial permitió modificar escala y geometría sin degradar la fuente durante la etapa de edición. Posteriormente, Adobe Photoshop se utilizó para aplicar las paletas cromáticas definidas para cada personaje, corregir detalles, preparar variaciones de pose y realizar los ajustes finales antes de la exportación.

4.1.1 Dirección de arte y propuesta visual

Mundo Aprendo utiliza una propuesta visual 2D dirigida a niños de 4 a 5 años. El estilo final emplea formas redondeadas, siluetas reconocibles, contrastes definidos y personajes con expresiones visibles. El menú principal y la selección de mundos mantienen una ambientación espacial, mientras que cada actividad introduce elementos relacionados con su temática: música, cuentos, comparación de tamaños y emociones.

La dirección de arte mantiene unidad entre personajes, escenarios, botones, iconos y demás elementos de interacción. Para conservar esa unidad se utiliza una paleta consistente, una composición sencilla y una jerarquía visual que diferencia los elementos interactivos de los fondos.

La propuesta visual se construyó a partir de una identidad común, pero no obligó a que todos los mundos utilizaran exactamente los mismos colores. Cada personaje conserva una paleta propia que permite reconocerlo dentro de su actividad, mientras que la interfaz

mantiene formas, bordes, estrellas, paneles y tratamientos de botón suficientemente consistentes para que el cambio de escena no parezca un cambio de aplicación.

La relación con el área de Educación se limitó a la conceptualización inicial de personajes, paletas y necesidades temáticas. La adaptación gráfica, la preparación técnica de los archivos y la decisión sobre cómo debían integrarse en escenas, paneles y estados de interfaz formaron parte del desarrollo artístico y tecnológico documentado en esta tesis.

La síntesis visual reúne los colores y formas que se repiten en el proyecto y funciona como referencia para revisar nuevos recursos antes de incorporarlos. Su finalidad es conservar la coherencia entre personajes, interfaz y ambientación sin convertir una paleta puntual en una restricción rígida para todos los mundos.

Ilustración 15

Síntesis de la identidad visual de Mundo Aprendo.

Nota. Elaboración propia a partir de los recursos gráficos de Mundo Aprendo.

4.1.2 Diseño de personajes, escenarios y elementos gráficos

El elenco visual incorpora a Luli como personaje principal y guía general, a Biblio el Búho en el Mundo de los Cuentos, al personaje explorador del Mundo de los Tamaños, a Nuna la Nube en el Mundo de las Emociones y a Tico el Tamborcito en el Mundo Musical.

Para cada personaje se prepararon vistas o poses acordes con su función: lectura, saludo, explicación, celebración, reintento o acompañamiento visual.

Las primeras referencias de personajes funcionaron como punto de partida y no como sprites definitivos. En Illustrator se reconstruyeron las formas y contornos para obtener una base editable y escalable; en Photoshop se aplicaron las paletas cromáticas y se completaron detalles de acabado. Esta separación entre forma y acabado facilitó corregir proporciones sin rehacer el color y modificar colores sin alterar la estructura principal del personaje.

El mismo criterio se utilizó con los escenarios. Las referencias iniciales ayudaron a establecer la temática de cada ambiente, pero los fondos se adaptaron antes de integrarse al proyecto para mantener una lectura clara de los elementos interactivos. La composición dejó zonas libres alrededor de paneles, botones y personajes con el fin de evitar que la ambientación compitiera visualmente con la consigna o con el objeto que debía seleccionarse.

La interfaz combinó recursos propios con elementos procedentes de los paquetes Cartoon UI, Hyper_Casual_UI y Space_Exploration_GUI_Kit. Estos recursos se utilizaron como base y fueron modificados en Photoshop e Illustrator cuando su color, proporción, forma o estilo no coincidían con la identidad de Mundo Aprendo. Por esta razón, el manual distingue entre creación propia, adaptación y procedencia del asset; antes de una distribución pública debe conservarse además la información de licencia correspondiente a cada paquete.

Ilustración 16

Referencia conceptual inicial utilizada para orientar la propuesta visual de mundos.

Nota. Referencia conceptual inicial proporcionada por el área de Educación y utilizada únicamente como guía para el desarrollo visual; no corresponde al arte final del videojuego.

La referencia conceptual inicial permitió reconocer una idea general de mundos diferenciados, personajes guía y una navegación visual mediante planetas. Durante el desarrollo esa propuesta se simplificó y se adaptó a las cuatro actividades finalmente implementadas. La imagen se conserva como evidencia de conceptualización y no representa una captura ni una composición final del videojuego.

A partir de esa referencia se realizaron bocetos y variaciones que permitieron separar el diseño conceptual del recurso técnico. En particular, Luli se trabajó en varias poses para cubrir estados de orientación, lectura y acompañamiento. Cada pose debía mantener la misma silueta, proporciones y paleta para que el cambio de sprite durante la ejecución se percibiera como una acción del mismo personaje y no como una ilustración distinta.

Ilustración 17

Bocetos de poses de Luli utilizados durante la definición del personaje.

Nota. Bocetos de trabajo utilizados como etapa previa a la reconstrucción y acabado del personaje.

Una vez definida la silueta, se prepararon las poses finales necesarias para el videojuego. La elección de poses respondió a estados concretos de la experiencia: presentar una instrucción, acompañar una lectura, celebrar un resultado o permanecer en reposo. El objetivo no fue producir animación cuadro a cuadro, sino disponer de variantes suficientemente claras para que los scripts pudieran cambiar el estado visual del personaje cuando la mecánica lo requiriera.

Ilustración 18

Variaciones finales de pose de Luli preparadas como recursos independientes.

Nota. Elaboración propia a partir de las variantes finales preparadas para la integración del personaje en Unity.

La separación por poses también simplificó la integración en Unity. Cada estado visual puede asignarse como un Sprite independiente a un componente Image o a un controlador de presentación, lo que evita recortar regiones de una lámina durante la

ejecución y permite cambiar una expresión mediante una referencia explícita desde el Inspector.

4.1.3 Producción de assets 2D

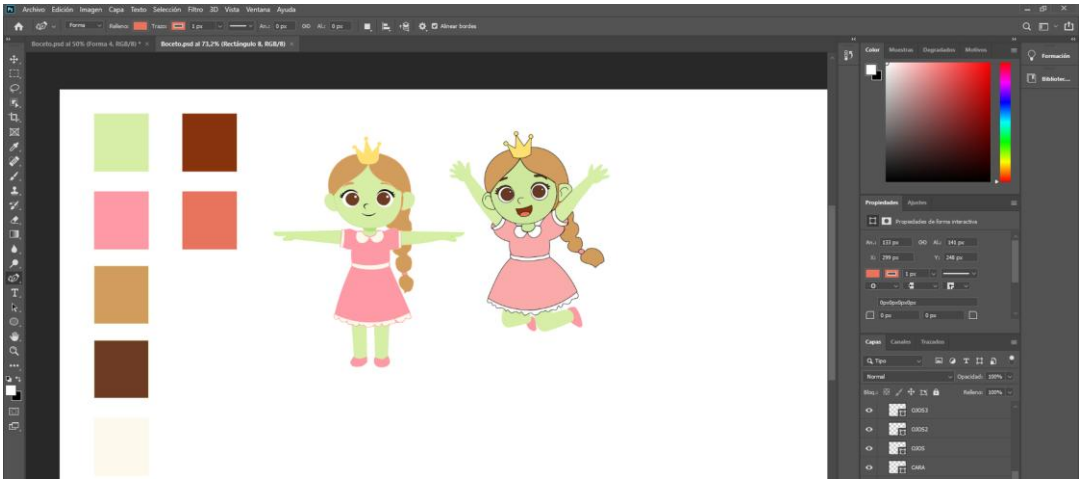
La producción de assets 2D siguió un flujo de edición dividido entre Illustrator y Photoshop. Illustrator se utilizó para construir o reconstruir las formas principales de los personajes y conservar un original vectorial editable. Photoshop se utilizó después para aplicar paletas, completar detalles, revisar bordes y preparar la composición final antes de convertir el recurso al formato requerido por Unity.

Los recursos terminados se exportaron principalmente como PNG con fondo transparente. La transparencia permite que personajes, iconos y botones se integren sobre fondos distintos sin incorporar un rectángulo de color alrededor del dibujo. Durante las primeras importaciones surgió un problema práctico: algunas láminas reunían varias poses en una misma imagen y resultaban incómodas para asignarlas directamente en las escenas. Para resolverlo, las poses se recortaron y exportaron de manera individual, de forma que cada archivo pudiera vincularse a un estado concreto del personaje dentro de Unity.

La etapa de acabado en Photoshop se utilizó también para verificar la paleta antes de exportar. En esta fase se corrigieron diferencias de tono, contornos y pequeños detalles que resultaban visibles al colocar el personaje sobre la interfaz. Mantener la paleta junto al personaje durante la edición permitió repetir colores con mayor consistencia entre poses.

Ilustración 19

Proceso de ajuste cromático y acabado de Luli en Adobe Photoshop.



Nota. Captura propia del proceso de edición y aplicación de paleta en Adobe Photoshop.

4.1.4 Pipeline de producción artística

El pipeline de producción artística parte de una referencia conceptual y termina únicamente cuando el recurso ha sido comprobado dentro de la escena. La primera etapa consiste en interpretar la referencia y decidir qué partes deben conservarse, simplificarse o adaptarse a la función del personaje o del fondo. Después se reconstruyen las formas en Illustrator y se prepara el acabado cromático en Photoshop. El archivo resultante se exporta a PNG y, cuando contiene varias poses, se divide en sprites independientes antes de su incorporación al proyecto.

La segunda mitad del pipeline ocurre dentro de Unity. El archivo se incorpora a la carpeta correspondiente, se importa como Sprite, se revisa la transparencia y se asigna al componente Image, al panel o al controlador que lo utilizará. La integración no se considera terminada con la importación: el recurso debe comprobarse en Game View para verificar escala, proporción, legibilidad y relación con los controles. Si el resultado no coincide con el diseño previsto, el flujo regresa a Photoshop o Illustrator y se genera una nueva versión.

Este flujo de ida y vuelta fue especialmente importante en la interfaz. Algunos recursos de Cartoon UI, Hyper_Casual_UI y Space_Exploration_GUI_Kit ofrecían una base funcional, pero sus colores o proporciones no coincidían con la propuesta visual. En esos casos se conservó únicamente la estructura útil del asset y se modificaron sus componentes gráficos antes de volver a importarlos, de manera que el recurso externo no determinara por sí solo la apariencia del proyecto.

La razón técnica para mantener archivos fuente editables y sprites finales separados es que cumplen funciones diferentes. Illustrator y Photoshop conservan la capacidad de modificación; Unity necesita archivos rasterizados listos para renderizar. Si se modifica únicamente el PNG final se pierde flexibilidad para cambios posteriores, mientras que conservar la fuente permite generar nuevamente el recurso en la resolución necesaria sin acumular degradación.

Ilustración 20

Trazabilidad de un recurso gráfico 2D: creación y exportación.

Nota. Elaboración propia a partir de los archivos y configuraciones conservados en el proyecto.

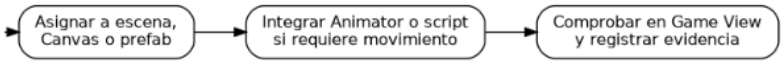
Ilustración 21

Trazabilidad de un recurso gráfico 2D: ubicación e importación en Unity.

Nota. Elaboración propia a partir de los archivos y configuraciones conservados en el proyecto.

Ilustración 22

Trazabilidad de un recurso gráfico 2D: configuración e integración final.



Nota. Elaboración propia a partir de los archivos y configuraciones conservados en el proyecto.

4.1.5 Preparación y animación de personajes y objetos

La animación del proyecto combina cambios de pose con movimientos programados de interfaz. Los personajes no utilizan rigging ni animación esquelética: cada reacción parte de sprites preparados previamente y el controlador de presentación sustituye la imagen o activa una pose según el momento de la actividad. Esta solución se eligió porque las acciones requeridas son breves y discretas —saludar, leer, celebrar, indicar un error o acompañar una instrucción— y no justificaban incorporar un sistema de huesos más complejo.

Se observaron controladores Animator asociados a los botones del menú y a sus estados Normal, Highlighted, Pressed, Selected y Disabled. La cantidad de controladores y clips debe confirmarse en la versión entregada antes de actualizar el inventario.

El proyecto también contiene animaciones programadas para transiciones de paneles y escenas, entradas escalonadas, aparición de estrellas, pulsaciones, movimientos flotantes, cambios de escala y retroalimentación mediante sacudidas o brillos.

Los scripts de presentación mantienen estas reacciones separadas de la lógica de la mecánica. Por ejemplo, MusicalLevelAnimationController administra poses y reacciones del personaje musical; StoryLevelAnimationController controla la entrada de paneles y la activación visual del guía del Mundo de los Cuentos; SizeWorldPresentationController reacciona a eventos del controlador de tamaños; y EmotionGameAnimationController concentra transiciones y movimientos reutilizables para el mundo de emociones. Esta separación permite modificar una animación sin alterar la regla que decide si la respuesta del usuario es correcta.

La misma idea se aplica a los botones y estrellas. Componentes como UIButtonFeedback, WorldCardAnimator, UIStarDisplay y RatingStarCelebration operan sobre escala, transparencia, posición o sprites de interfaz, mientras que los controladores de las actividades solo comunican el estado que debe representarse. La mecánica conserva así la responsabilidad de decidir y el componente visual la responsabilidad de mostrar.

Las variaciones de Biblio muestran el tipo de recurso preparado para este sistema de estados visuales. Las poses de lectura, explicación y celebración pueden asignarse a momentos distintos del Mundo de los Cuentos sin necesidad de reconstruir el personaje durante la ejecución.

Ilustración 23

Variaciones de pose de Biblio el Búho utilizadas como estados visuales del personaje.

Nota. Elaboración propia a partir de las variantes de pose preparadas para el Mundo de los Cuentos.

4.1.6 Configuración de sprites y renderización 2D

La integración gráfica se apoya en Universal Render Pipeline, en el sistema de interfaz de usuario de Unity y en TextMesh Pro. Las imágenes con transparencia se asignan a componentes Image y a los demás elementos de interfaz, conservando el canal alfa cuando corresponde.

No se aplicaron procesos de creación de materiales tridimensionales, mapeado de texturas sobre malla, configuración de fuentes de luz ni renderización de escenas 3D, debido a que el proyecto se resolvió íntegramente con recursos 2D. La apariencia final de cada elemento depende del archivo de imagen exportado y de su configuración de

importación como sprite, no de un material o de una iluminación calculada en tiempo de ejecución.

Aunque la fuente de algunos personajes se trabajó de forma vectorial, Unity no utiliza directamente esos documentos de Illustrator dentro de las escenas. El recurso final se rasteriza al exportarse como PNG y se maneja como Sprite. El beneficio de conservar la fuente vectorial se encuentra, por tanto, en la etapa de producción: permite corregir tamaño y forma con mayor libertad antes de generar una nueva versión rasterizada.

La transparencia del PNG fue relevante para la superposición de personajes sobre fondos y paneles. Un borde mal recortado o un canal alfa incorrecto se vuelve visible inmediatamente al cambiar el fondo de una escena; por ello, la comprobación del recurso debía realizarse sobre el contexto real de uso y no únicamente sobre el lienzo de Photoshop.

4.1.7 Creación de efectos visuales

Los efectos visuales comprobados en el prototipo se producen mediante cambios de escala, variaciones de transparencia, cambios de color, brillo, desplazamiento y pulsación aplicados sobre elementos de interfaz y sobre sprites. Estos efectos acompañan la retroalimentación de las actividades, la aparición de las estrellas y las transiciones entre paneles.

No se registraron sistemas de partículas, shaders personalizados ni efectos de posprocesamiento en la versión documentada. En consecuencia, este apartado se limita a los recursos efectivamente implementados y no describe técnicas que no formaron parte del desarrollo.

La elección de efectos basados en transformaciones de UI reduce la cantidad de recursos adicionales que deben mantenerse. Un pulso de escala, un cambio de alpha o un

desplazamiento puede aplicarse sobre el mismo sprite ya utilizado por la interfaz y configurarse mediante parámetros serializados. Esto resultó suficiente para destacar botones, indicar una selección, mostrar un acierto o introducir una pantalla sin incorporar un sistema visual independiente para cada caso.

4.1.8 Integración de recursos visuales en el motor

La integración visual se comprueba desde la ubicación del archivo en Project hasta su asignación en el Inspector, su presencia en la jerarquía y su resultado en Game View. El recurso gráfico por sí solo no produce comportamiento: debe vincularse a un componente de Unity y, cuando corresponde, quedar referenciado por un script. Un personaje puede estar asignado a un Image, una estrella a UIStarDisplay, una tarjeta a WorldCardView y una pose a un controlador de presentación. La relación entre asset, componente y script es la que convierte una imagen estática en parte del funcionamiento del videojuego.

Esta integración explica también por qué las poses se separaron en archivos independientes. Los controladores de presentación trabajan con referencias directas a sprites concretos; si varias poses permanecieran agrupadas en una sola lámina, sería necesario configurar un recorte adicional o crear un atlas con regiones específicas. Para el alcance del proyecto resultó más simple y mantenible asignar cada pose como un recurso individual y cambiar la referencia cuando la actividad requiere una reacción distinta.

Cuando se sustituye un recurso, la revisión debe incluir no solo la imagen final, sino también sus referencias. Un archivo con dimensiones muy diferentes puede conservar el enlace del Inspector y, aun así, alterar la composición; del mismo modo, renombrar o mover un asset desde fuera del editor puede perder la relación que Unity mantiene mediante

su archivo de metadatos. Por ello, los cambios de recursos deben realizarse desde el proyecto y comprobarse en la escena que los consume.

4.1.9 Optimización gráfica

La optimización gráfica comienza antes de la importación. Conservar una fuente vectorial en Illustrator permite generar el PNG con una resolución apropiada para su uso real, en lugar de ampliar una imagen rasterizada pequeña y producir bordes pixelados. En Photoshop se recorta el espacio transparente innecesario y se revisa que cada pose conserve una caja de imagen razonable, ya que un lienzo excesivamente grande aumenta memoria y dificulta la alineación en la interfaz.

Dentro de Unity, la revisión se concentra en el tamaño máximo, filtrado, compresión y uso de transparencia. No se establece un único valor para todos los sprites porque un fondo de pantalla y un icono pequeño tienen necesidades diferentes. El criterio consiste en mantener suficiente resolución para la escala visible sin conservar información que nunca se mostrará en pantalla.

En la interfaz, Canvas Scaler y los anclajes controlan la adaptación a las resoluciones documentadas. La comprobación visual debe observar especialmente elementos cercanos a los bordes, textos, tarjetas y personajes, porque son los componentes que evidencian primero una relación de aspecto distinta. Esta verificación corresponde a estabilidad visual y no a rendimiento de GPU.

4.1.10 Software, formatos y organización de recursos

La producción artística utilizó Adobe Illustrator para la construcción y adaptación de formas vectoriales y Adobe Photoshop para color, acabado, composición y preparación de archivos finales. Los recursos destinados al motor se exportaron principalmente como

PNG por su compatibilidad con transparencia; los documentos fuente de edición se conservaron fuera de la lógica de ejecución para permitir modificaciones posteriores.

La interfaz incorpora además recursos adaptados a partir de Cartoon UI, Hyper_Casual_UI y Space_Exploration_GUI_Kit. Estos paquetes no se documentan como producción propia: su aporte consiste en servir como base para determinados botones, tarjetas, paneles o elementos gráficos que fueron modificados para coincidir con la paleta y la composición del proyecto. La procedencia debe conservarse junto con el inventario de recursos y revisarse frente a las condiciones de licencia antes de cualquier distribución pública.

La organización de archivos sigue la estructura de carpetas del proyecto y separa personajes, fondos, interfaz, animales y animaciones. La nomenclatura estable resulta importante porque los recursos se relacionan con escenas y componentes mediante referencias del editor; por ello, un cambio de nombre o ubicación debe realizarse dentro de Unity y comprobarse inmediatamente en las escenas afectadas.

4.1.11 Validación técnica de recursos visuales

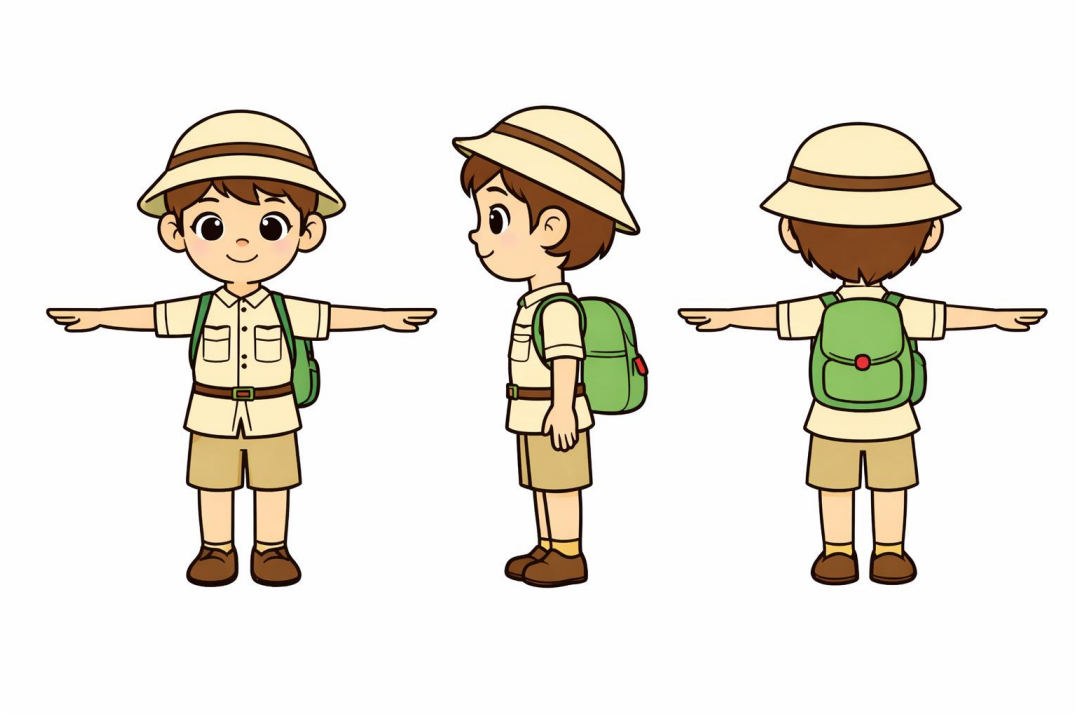
La validación técnica de un recurso visual se realiza en dos niveles. El primero revisa el archivo aislado: transparencia, contorno, consistencia cromática, proporción entre poses y ausencia de bordes recortados. La segunda comprueba el recurso dentro de la escena: escala, posición, relación con el fondo, lectura junto a textos y botones y comportamiento al cambiar el tamaño de la ventana o la resolución de ejecución.

Las hojas de vistas se utilizaron como control interno de proporciones antes de preparar variantes de pose. El personaje guía conserva rasgos, vestuario y accesorios comunes entre la vista frontal, lateral y posterior; esta coherencia facilita crear nuevas poses

sin alterar su identidad visual. En la validación dentro de Unity, el mismo criterio se traslada a la comparación entre sprites que se intercambian durante una escena.

Ilustración 24

Vistas frontal, lateral y posterior del personaje guía utilizadas para controlar proporciones.



Nota. Elaboración propia a partir de la hoja de vistas utilizada para mantener proporciones del personaje guía.

4.1.12 Procedimiento para ampliar recursos artísticos

Para incorporar un recurso artístico se debe respetar el estilo visual, exportarlo con un nombre descriptivo y ubicarlo en la carpeta correspondiente. Luego se importa como Sprite, se compara su configuración con recursos equivalentes y se asigna a la escena o al Canvas. Si requiere movimiento, se conecta con el Animator o el script responsable; la comprobación final cubre proporción, transparencia, estados interactivos y adaptación del Canvas.

La ampliación debe actualizar el inventario de recursos, la licencia o procedencia, la configuración de importación, la escena de uso y la evidencia visual correspondiente. Un

archivo no debe considerarse integrado mientras no se compruebe dentro de la versión entregada.

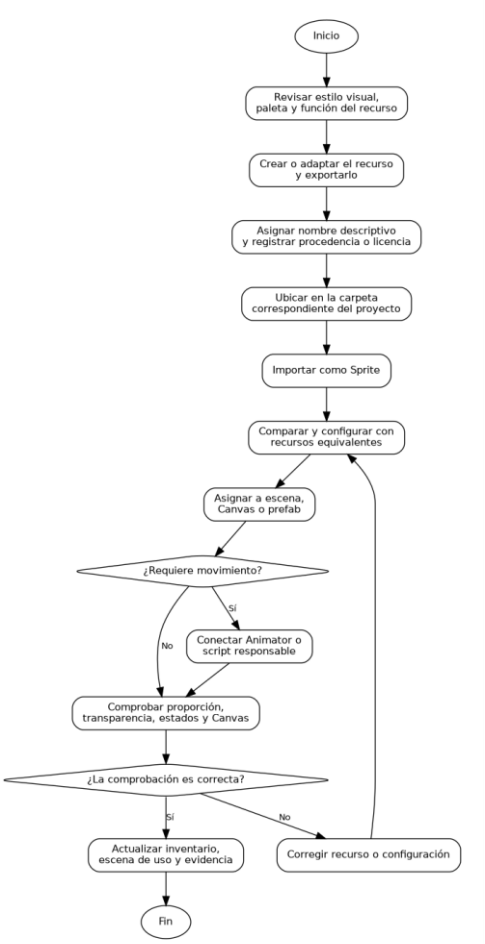
Si el nuevo recurso parte de una referencia conceptual, esta debe tratarse como una guía y no como el archivo final. El procedimiento recomendado consiste en reconstruir o adaptar primero la forma en Illustrator, completar color y detalle en Photoshop, exportar el PNG y separar las variantes de pose que deban cambiar de forma independiente durante la ejecución. Después se incorpora el archivo desde Unity y se asigna a la escena o componente responsable.

Si el recurso procede de un paquete externo, la ampliación debe registrar el paquete de origen y comprobar que la licencia permita el uso previsto. Cualquier modificación de color, borde, proporción o composición debe conservarse como una versión propia del proyecto para evitar sobrescribir el original y para que futuras actualizaciones del paquete no eliminen los ajustes realizados.

El diagrama debe ordenar el procedimiento desde la revisión del estilo, la exportación y la nomenclatura hasta la carpeta de destino, la configuración como Sprite, la asignación, la conexión con Animator o script cuando corresponda y la comprobación final en Game View.

Ilustración 25

Procedimiento para incorporar un nuevo recurso artístico en Unity.



Nota. Elaboración propia a partir del procedimiento técnico documentado.

4.2 Manual del programador

Este bloque documenta la construcción técnica del prototipo desde la perspectiva de quien debe mantenerlo o ampliarlo. Presenta la arquitectura general, la organización del proyecto, las clases y componentes, el desarrollo de los sistemas y de las mecánicas, la persistencia, la configuración de escenas, las pruebas realizadas, el despliegue y los procedimientos de escalabilidad.

4.2.1 Arquitectura general del sistema

La arquitectura de Mundo Aprendo se organizó en tres capas con responsabilidades separadas: una capa de servicios transversales, una capa de controladores de actividad y una capa de componentes de presentación. Esta separación respondió a una condición concreta del proyecto: los cuatro mundos comparten comportamientos idénticos, como registrar estrellas, cambiar de escena o reproducir un sonido, pero difieren por completo en su mecánica. Concentrar lo común en servicios reutilizables evitó repetir esa lógica cuatro veces y permitió que la incorporación de un mundo no obligara a modificar los anteriores.

La capa de servicios transversales agrupa el comportamiento que no pertenece a ninguna actividad concreta. La componen clases estáticas y componentes persistentes que no conservan estado de escena: la navegación, el repositorio de progreso, el cálculo de estrellas, la transición visual entre escenas y la gestión de audio. Al ser estáticas, estas clases resultan accesibles desde cualquier escena sin necesidad de asignar referencias en el Inspector, lo que eliminó una fuente frecuente de referencias nulas durante la carga de escenas.

La capa de controladores de actividad contiene un componente principal por cada mundo temático. Cada controlador es el único responsable del flujo de su actividad: prepara el estado inicial, habilita la entrada del usuario, valida la respuesta, calcula el resultado y solicita su registro. Ningún controlador conoce a los demás; los cuatro se comunican exclusivamente con la capa de servicios.

La capa de componentes de presentación reúne elementos reutilizables de interfaz que no deciden nada sobre la actividad y se limitan a mostrar el estado que reciben: la visualización de estrellas, las transiciones de panel, la barra de progreso, la

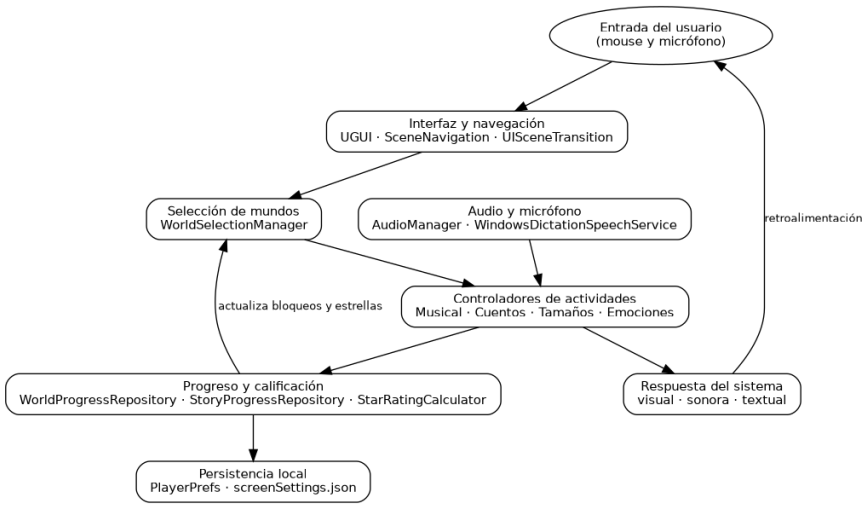
retroalimentación de los botones y las animaciones de entrada. Su carácter pasivo permitió emplearlos sin adaptaciones tanto en las cuatro escenas de juego como en la pantalla de selección.

De ese reparto se deriva la regla de dependencia que ordena el sistema: las capas superiores conocen a las inferiores, pero no a la inversa. Un controlador de mundo invoca al repositorio de progreso, mientras que el repositorio desconoce qué mundo lo invocó y opera únicamente sobre un índice numérico. Del mismo modo, los componentes de presentación reciben valores ya calculados y no acceden a la lógica de las actividades. Esta orientación de las dependencias es la que permite modificar una mecánica sin afectar a la persistencia, y modificar la persistencia sin revisar las cuatro mecánicas.

La Ilustración 26 representa esta organización y las relaciones entre las tres capas.

Ilustración 26

Arquitectura modular de Mundo Aprendo.



Nota. Elaboración propia.

En el esquema se observa que el flujo de una interacción sigue siempre la misma dirección: el evento de interfaz llega al controlador de la escena, este valida la entrada

según la regla de su actividad, actualiza la respuesta visual y sonora, al concluir la actividad, solicita el registro del resultado al repositorio de progreso. Ese registro constituye el único punto del sistema en el que un mundo escribe información persistente.

4.2.2 Tecnologías, motores y herramientas utilizadas

El manual corresponde a la versión de Mundo Aprendo desarrollada en Unity para Windows de 64 bits. El proyecto integra las escenas Menu, SeleccionMundos, MundoMusical, MundoCuentos, MundoTamanos y MundoEmociones, con interacción por mouse, audio, progreso local y un módulo de lectura apoyada por voz.

Para revisar o mantener la aplicación se requiere conservar la carpeta completa del proyecto o de la compilación, verificar la salida de audio y, cuando se pruebe el módulo de Cuentos, habilitar un micrófono y los permisos de voz de Windows. La Tabla 15 resume las condiciones de ejecución; la Ilustración 27 identifica la pantalla de inicio utilizada como punto de comprobación.

Tabla 15

Condiciones técnicas y de comprobación de Mundo Aprendo.

Elemento	Requisito o condición documentada
Sistema operativo	Windows de 64 bits.
Pantalla	Resolución configurada de 1024 × 768; modo de pantalla completa y ventana no redimensionable.
Resoluciones previstas para comprobación	1920 × 1080, 1366 × 768 y 1280 × 720, definidas para revisar la permanencia de los controles dentro del Canvas.
Dispositivo de entrada	Mouse para seleccionar botones, opciones y respuestas.
Audio	Parlantes o audífonos para música, efectos e instrucciones sonoras.
Micrófono	Requerido para el Mundo de Cuentos; depende de permisos y servicios de voz de Windows.
Distribución	Ejecutar Mundo aprendo tesis.exe sin separarlo de la carpeta Data ni de las bibliotecas generadas por Unity.

Nota. Elaboración propia.

Las versiones y finalidades verificadas se concentran en la Tabla 16. Unity se utiliza como motor y editor; C# implementa la lógica; URP, UGUI y TextMesh Pro resuelven la presentación y la interfaz; Windows Standalone define el destino y condiciona el servicio de dictado. La Ilustración 27 identifica la versión del editor con la que debe abrirse el proyecto.

Tabla 16

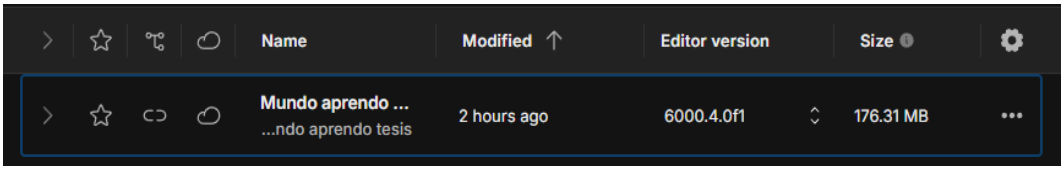
Tecnologías y herramientas verificadas.

Tecnología o recurso	Versión o configuración	Uso verificado
Unity	6000.4.0f1	Motor de escenas, interfaz, recursos y minijuegos 2D.
Plataforma	Windows Standalone x86-64	Ejecución y compilación de escritorio.
Backend	Mono	Ensamblados administrados y carpeta MonoBleedingEdge.
Universal Render Pipeline	17.4.0	Pipeline gráfico del proyecto.
Unity Input System	1.19.0	Infraestructura de entrada instalada.
Unity Test Framework	1.6.0	Ejecuciones EditMode y PlayMode.
UGUI / TextMesh Pro	2.0.0	Interfaz y texto del videojuego.
Voz	DictationRecognizer y UnityEngine.Microphone	Lectura apoyada por voz en Windows.

Nota. Elaboración propia.

Ilustración 27

Versión de Unity utilizada para abrir el proyecto.



Nota. Captura propia del proyecto y de sus registros técnicos.

4.2.3 Organización técnica del proyecto

La organización de carpetas del proyecto siguió un criterio doble: el código se agrupó por función y el código propio de cada actividad se agrupó por mundo. Esta distinción permite localizar un archivo a partir de una pregunta simple, ya que si el comportamiento afecta a todo el videojuego se busca por función, y si pertenece a una sola actividad se busca por el nombre del mundo.

Las carpetas de función reúnen los servicios compartidos. Navigation concentra los nombres de escena y el cargador; Progress agrupa el repositorio de progreso y el cálculo de estrellas; UI contiene los componentes reutilizables de interfaz; Options reúne la configuración de audio, vídeo y micrófono; y WorldSelection agrupa los componentes de la pantalla de selección de mundos.

Las carpetas de mundo contienen los controladores y componentes propios de cada actividad, y se denominan MundoMusical, MundoCuentos, MundoTamanos y MundoEmociones. Cada una es autónoma, ya que sus archivos no se referencian desde otro mundo; retirar una carpeta completa no impide compilar las restantes.

Dentro de dos de esas agrupaciones existe una subcarpeta denominada Editor. Su nombre no es descriptivo sino funcional: Unity excluye automáticamente de la compilación final cualquier script situado en una carpeta con ese nombre. Allí se ubicaron las herramientas que generan escenas desde el menú del editor, útiles durante el desarrollo pero que no deben formar parte del ejecutable distribuido.

El conjunto del código comprende 61 archivos de C# y 12 232 líneas. Su distribución no es homogénea y refleja la complejidad real de cada parte: el mundo de los cuentos concentra la mayor cantidad de código por incorporar el reconocimiento de voz,

seguido por el mundo musical; las carpetas de interfaz y de opciones ocupan el tercer y cuarto lugar por reunir componentes reutilizables; y las carpetas de navegación y de progreso son las más reducidas, precisamente porque su responsabilidad está acotada a una única función.

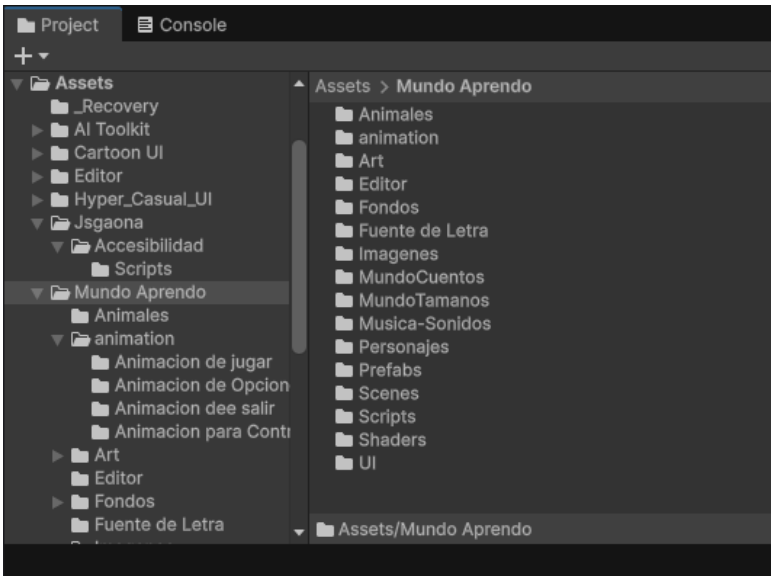
Las convenciones de nombres siguen el rol de cada componente. Los sufijos Controller y Manager identifican a las clases que gobiernan el flujo de una escena o de una actividad; Repository, a las que conservan y recuperan datos; View, a las que únicamente muestran información; y Animator o AnimationController, a las que ejecutan efectos visuales. Un archivo puede así reconocerse por su nombre antes de abrirlo.

Todo el código se agrupó bajo espacios de nombres asociados al proyecto, lo que reduce colisiones con paquetes externos y hace explícito qué clases pertenecen a Mundo Aprendo; las herramientas del editor utilizan espacios de nombres derivados. Esta organización facilita localizar controladores, repositorios, servicios y utilidades desde el código y mantiene separadas las responsabilidades propias del juego respecto de las extensiones utilizadas únicamente durante el desarrollo.

Las Ilustraciones 28, 29 y 30 muestran esta organización tal como se presenta en el editor de Unity.

Ilustración 28

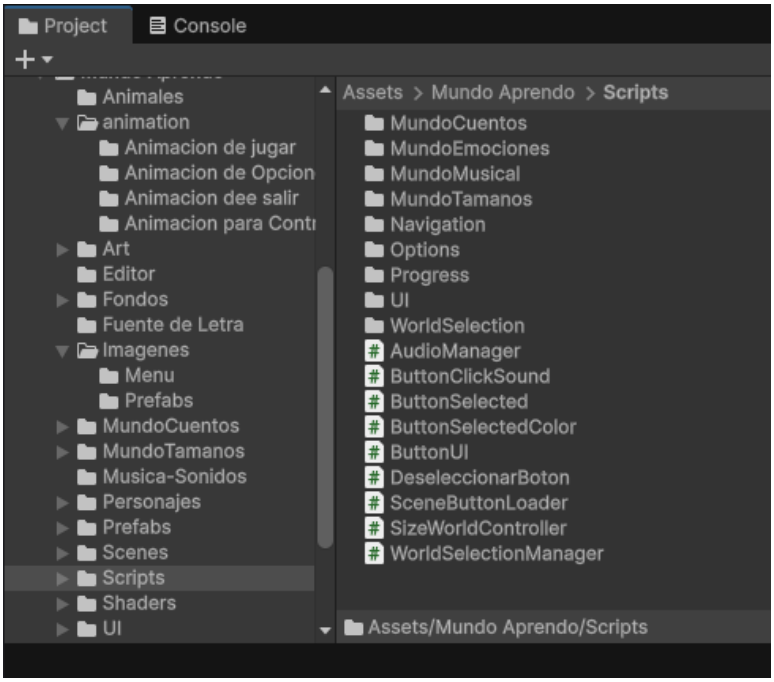
Organización general de carpetas del proyecto.



Nota. Captura propia del proyecto y de sus registros técnicos.

Ilustración 29

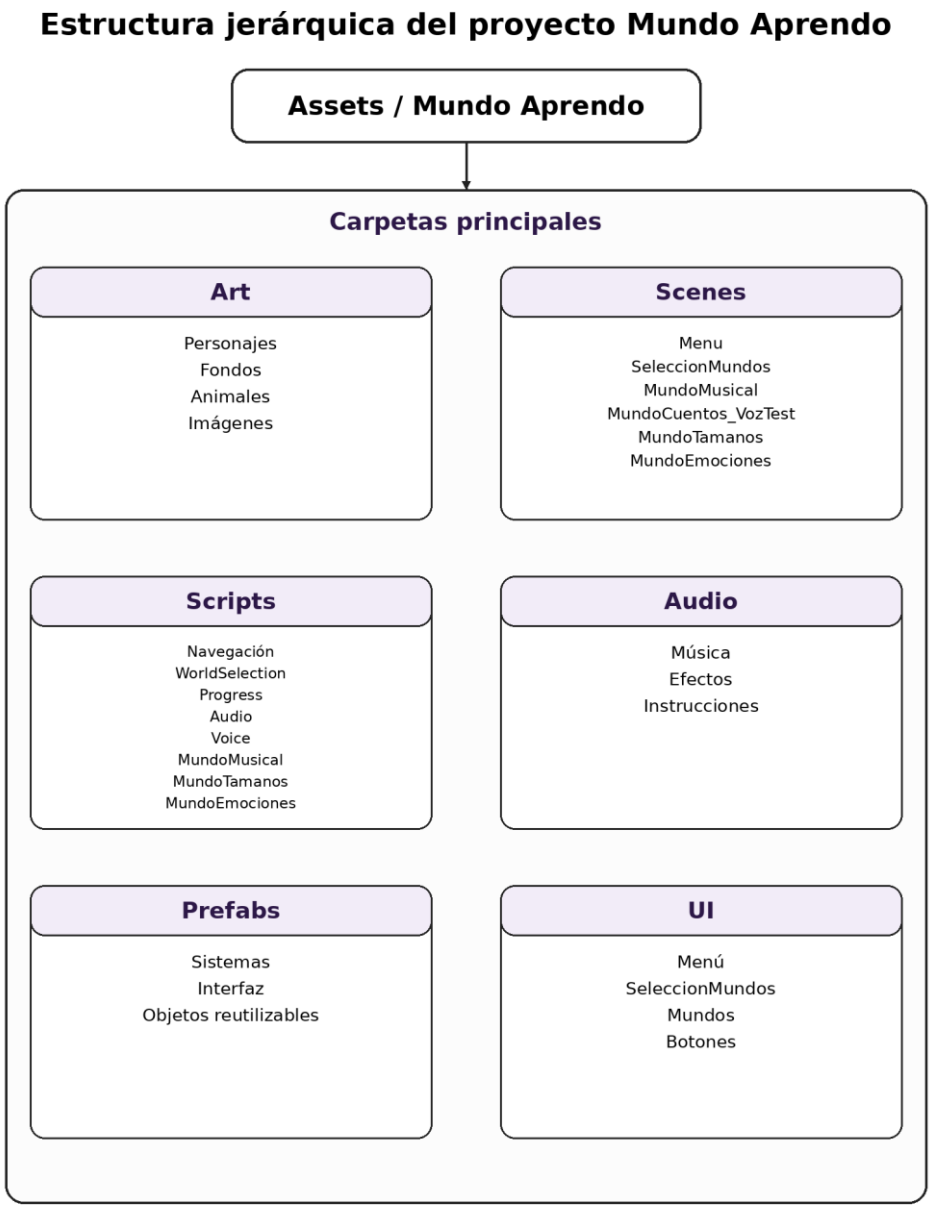
Carpeta Scripts del proyecto.



Nota. Captura propia del proyecto y de sus registros técnicos.

Ilustración 30

Estructura jerárquica del proyecto Mundo Aprendo.



Nota. Captura propia del proyecto y de sus registros técnicos.

4.2.4 Arquitectura de clases y componentes

Las clases del proyecto se agrupan en tres categorías según su relación con el ciclo de vida de Unity. La primera reúne clases estáticas que no se instancian ni dependen de un

objeto de escena, y que concentran los servicios transversales. La segunda agrupa los componentes que se adjuntan a un objeto de la escena y gobiernan el comportamiento de una actividad. La tercera corresponde a componentes también adjuntos a objetos de escena, pero sin capacidad de decisión sobre la actividad, dedicados exclusivamente a la presentación.

El acceso a los datos persistentes se resolvió mediante el patrón repositorio. En lugar de que cada mundo escriba directamente en el almacenamiento local, un único componente concentra las operaciones de lectura y escritura y expone métodos con significado propio del dominio, como consultar las estrellas de un mundo, comprobar si está desbloqueado o registrar el mejor resultado. Las claves de almacenamiento se construyen dentro del repositorio a partir de un formato único y ningún otro archivo del proyecto las escribe de forma literal. Esa restricción tiene un efecto práctico inmediato: modificar el formato de las claves obliga a intervenir un solo archivo.

La comunicación entre el repositorio de progreso y la interfaz se resolvió mediante un evento. Cuando el repositorio registra un cambio, notifica a quien esté suscrito sin conocer su identidad. Cuatro componentes de la pantalla de selección se suscriben a esa notificación al activarse y cancelan la suscripción al desactivarse: el gestor de la selección, la vista de cada tarjeta de mundo, el panel de estrellas y los efectos visuales de la pantalla. Gracias a ese mecanismo, completar una actividad actualiza los bloqueos y las estrellas de la selección sin necesidad de recargar la escena y sin que el repositorio contenga una sola referencia a la interfaz.

El reconocimiento de voz se aisló detrás de una interfaz. El controlador del mundo de los cuentos no invoca directamente el motor de dictado del sistema operativo, sino un

contrato que declara las operaciones de iniciar y detener la escucha junto con los eventos de resultado parcial, resultado final, error y cambio de estado. La implementación concreta para Windows cumple ese contrato. La separación tiene una consecuencia precisa para el mantenimiento: sustituir el motor de reconocimiento por otro exige escribir una nueva implementación del contrato, pero no modificar el controlador de la actividad ni su interfaz de usuario.

La Tabla 17 relaciona los componentes principales del proyecto con su área, su responsabilidad y el componente con el que se comunica.

Tabla 17

Clases y componentes principales verificados.

Script o componente	Área asociada	Responsabilidad	Relación principal
MundoAprendoSceneNames / SceneNavigation	Navegación	Centralizar los nombres de escena, validar y cargar.	UISceneTransition
UISceneTransition	Navegación	Fundido de salida y entrada entre escenas.	SceneNavigation
WorldProgressRepository	Progreso	Guardar el mejor resultado y notificar cambios.	PlayerPrefs y suscriptores
StarRatingCalculator	Progreso	Traducir el número de errores a estrellas.	Controladores de mundo
AudioManager	Audio	Música, efectos y volúmenes conservados.	AudioOptionsController
WorldSelectionManager	Selección de mundos	Tarjetas, bloqueos, estrellas y reinicio.	WorldProgressRepository
WorldCardView / WorldStarsPanelView	Selección de mundos	Estado visual de cada tarjeta y del panel.	WorldProgressRepository
WorldSelectionVisuals	Selección de mundos	Efectos visuales según el desbloqueo.	WorldProgressRepository
MundoMusicalSequenceGame	Mundo Musical	Reproducir y validar secuencias musicales.	UI, audio y progreso

Script o componente	Área asociada	Responsabilidad	Relación principal
MusicalKeyVisual	Mundo Musical	Estado visual de cada tecla del piano.	MundoMusicalSequenceGame
MusicalTutorialController	Mundo Musical	Tutorial inicial y su estado conservado.	MundoMusicalSequenceGame
VoiceRecognitionTest	Mundo de los Cuentos	Coordinar cuento, escucha, validación y resultado.	ISpeechToTextService
ISpeechToTextService	Módulo de voz	Contrato de reconocimiento independiente del motor.	WindowsDictationSpeechService
WindowsDictationSpeechService	Módulo de voz	Implementación del contrato con DictationRecognizer.	ISpeechToTextService
ReadingEvaluator	Módulo de voz	Normalizar el texto y calcular la similitud.	VoiceRecognitionTest
StoryProgressRepository	Mundo de los Cuentos	Progreso por cuento y estado del tutorial.	WorldProgressRepository
SizeWorldController	Mundo de los Tamaños	Gestionar rondas y selección por tamaño.	UI y progreso
SizeWorldTutorialController	Mundo de los Tamaños	Tutorial inicial previo a la actividad.	SizeWorldController
EmotionGameManager	Mundo de las Emociones	Gestionar rondas y selección de emoción.	StarRatingCalculator
EmotionAnswerButton / EmotionResultPanel	Mundo de las Emociones	Entrega de la respuesta y panel de cierre.	EmotionGameManager
CharacterDialogueController	Interfaz	Personaje guía, diálogos y retroalimentación.	GameplayDialogueFeedbackBridge
GameplayDialogueFeedbackBridge	Integración	Conectar el personaje guía a los cuatro mundos.	Eventos de los controladores
UIStarDisplay / UIPanelTransition / UIProgressBar	Interfaz	Mostrar estrellas, transiciones y avance.	Controladores de mundo
MicrophoneSettings / AudioOptionsController	Opciones	Selección y prueba de micrófono; volúmenes.	PlayerPrefs
ManagerVideo	Opciones	Resolución, modo de pantalla y sincronización.	screenSettings.json

Script o componente	Área asociada	Responsabilidad	Relación principal
WorldSelectionSceneBuilder / SizeWorldSceneBuilder	Herramientas de editor	Generar escenas desde el menú del editor.	Excluidos del ejecutable

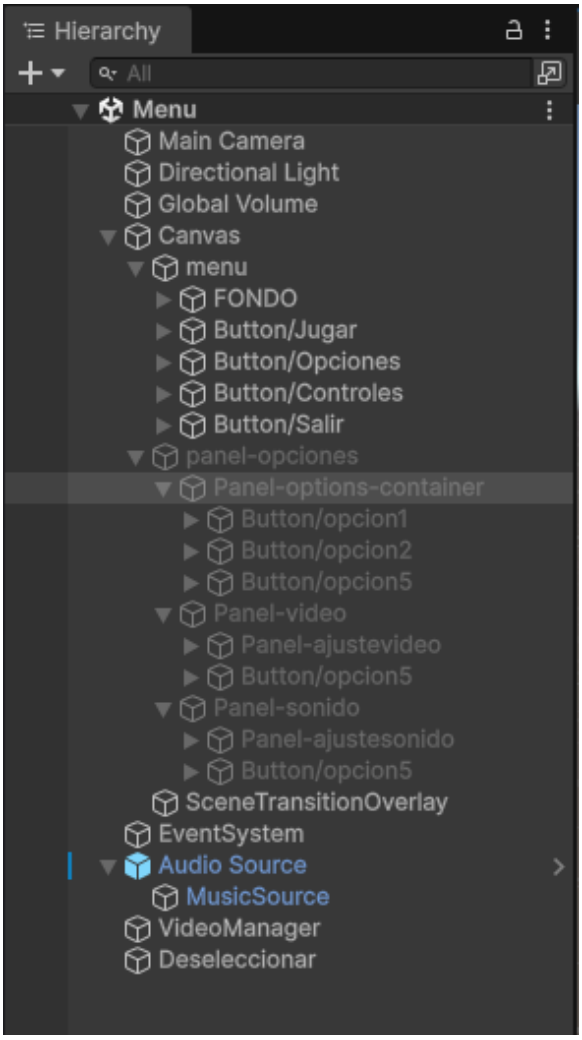
Nota. Elaboración propia.

La lectura de la tabla permite reconstruir las dependencias del sistema. Los componentes de las cinco primeras filas son transversales y no pertenecen a ninguna actividad concreta; los controladores de mundo dependen de ellos, pero no entre sí; y los componentes de presentación aparecen siempre como destino de una relación, nunca como origen. Los dos generadores de escena constituyen un caso aparte, ya que se ejecutan únicamente dentro del editor y no forman parte del videojuego compilado.

Para intervenir una escena conviene seguir ese mismo orden de lectura: identificar primero qué controlador recibe los eventos de la interfaz, después qué repositorio conserva el estado afectado y, por último, qué servicio externo interviene, lo que resulta especialmente relevante en audio y en voz. Las Ilustraciones 31 y 32 muestran cómo se refleja esta organización en la jerarquía de objetos de dos escenas del proyecto.

Ilustración 31

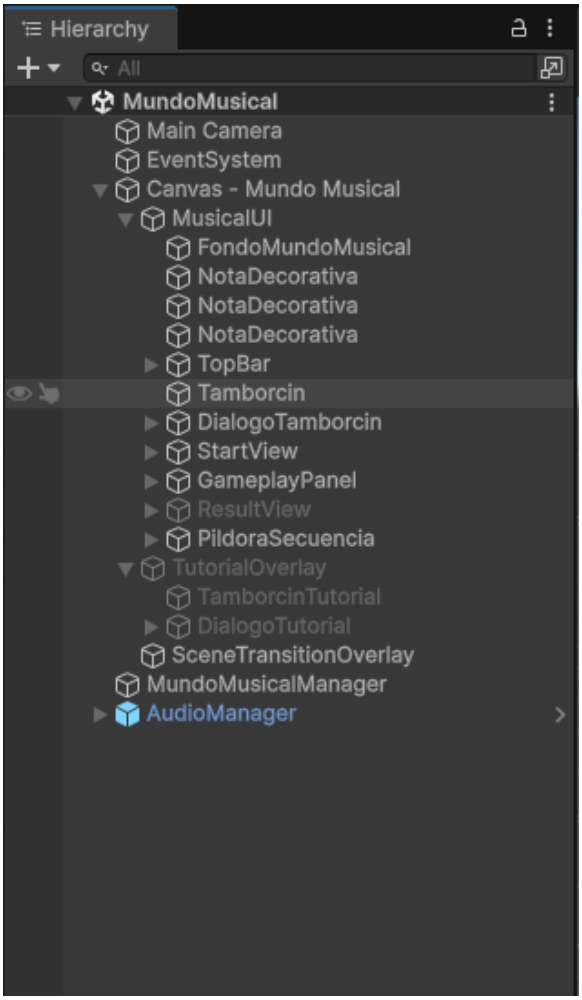
Jerarquía del menú principal.



Nota. Captura propia del proyecto y de sus registros técnicos.

Ilustración 32

Jerarquía de la escena Mundo Musical.



Nota. Captura propia del proyecto y de sus registros técnicos.

Los apartados siguientes describen uno por uno los componentes recogidos en la Tabla 17, en el mismo orden en que aparecen en ella. Cada uno indica qué resuelve el componente, dónde reside dentro del proyecto, qué recibe como entrada, qué produce y por qué se resolvió de esa manera y no de otra. La descripción se limita a la responsabilidad de cada archivo y a su relación con los demás; Esta organización responde a una necesidad concreta de mantenimiento quien deba intervenir el proyecto necesita localizar primero el

archivo responsable de un comportamiento antes de seguir el flujo completo en el que participa.

4.2.4.1 MundoAprendoSceneNames y SceneNavigation

Estos dos archivos forman la única puerta de entrada a cualquier cambio de pantalla del videojuego. MundoAprendoSceneNames no contiene lógica: declara como constantes los nombres de las seis escenas registradas en la configuración de compilación y existe para que ningún otro archivo del proyecto escriba un nombre de escena de forma literal. SceneNavigation es el componente que recibe la solicitud de carga, comprueba que el destino esté habilitado en la configuración de compilación y ordena la apertura de la escena. La entrada que recibe procede siempre de un evento de clic de la interfaz o de un panel de cierre de actividad, y su salida es el cambio de escena efectivo.

La separación entre el nombre y la carga responde a un problema comprobado durante el desarrollo. Un nombre de escena escrito directamente dentro de un botón o de un controlador es una cadena de texto que el compilador no verifica, de modo que un error ortográfico o un cambio de nombre no produce ningún aviso y solo se manifiesta cuando el usuario pulsa el botón. Al declararse en un único archivo, renombrar la escena de cuentos exigió modificar solamente la constante y todas las transiciones que apuntaban a ella siguieron funcionando sin tocar ningún otro archivo.

SceneNavigation tampoco ejecuta el efecto visual del cambio: lo delega en UISceneTransition antes de abrir la escena destino. Esta división mantiene en componentes distintos dos decisiones que no tienen relación entre sí, que son qué pantalla debe cargarse y cómo debe verse ese cambio. La consecuencia práctica es que el efecto de transición puede sustituirse sin revisar las rutas y una ruta puede modificarse sin reprogramar la

animación. El recorrido completo desde el clic hasta la escena cargada se representa en la Ilustración 39.

4.2.4.2 UISceneTransition

UISceneTransition resuelve la parte visible del cambio de escena mediante un fundido de salida antes de la carga y un fundido de entrada una vez abierta la escena nueva. Se trata de un componente de presentación en sentido estricto: no decide a qué pantalla se va, no consulta el progreso y no conoce las reglas de desbloqueo. Su única entrada es la orden que le entrega SceneNavigation y su única salida es la opacidad de una capa que cubre toda la pantalla durante el intervalo en que la escena anterior desaparece y la siguiente todavía no se ha dibujado.

El motivo de que este componente exista de forma independiente es que la carga de una escena en Unity no es instantánea y, sin una capa intermedia, el usuario percibe un salto brusco entre dos imágenes sin relación. Al cubrir ese intervalo con un fundido, el cambio se lee como una continuidad y no como un corte. Dado que la duración del efecto está expuesta como parámetro, ajustar la sensación de rapidez del videojuego no exige modificar ninguna línea de la lógica de navegación.

4.2.4.3 WorldProgressRepository

WorldProgressRepository concentra toda la persistencia del avance de los cuatro mundos y es el único archivo del proyecto autorizado a leer y escribir esas claves. Está implementado como clase estática, por lo que no necesita un objeto propio en cada escena ni depende de que alguien lo instancie antes de usarlo. Recibe como entrada un índice de mundo comprendido entre cero y tres junto con un número de estrellas, y expone hacia el resto del proyecto operaciones con significado propio del dominio, como consultar las

estrellas de un mundo, comprobar si está desbloqueado o registrar un resultado. Su funcionamiento detallado, incluidas las claves empleadas y el procedimiento de reinicio, se documenta en el apartado 4.2.9.

La decisión de aplicar el patrón repositorio tiene un efecto directo sobre el mantenimiento. Si cada mundo escribiera por su cuenta en el almacenamiento local, el formato de las claves quedaría repartido entre cuatro controladores y cualquier cambio obligaría a revisarlos todos, con el riesgo de que uno quedara desincronizado y produjera un avance que el resto del sistema no reconoce. Al construirse las claves dentro del repositorio a partir de un formato único, modificar ese formato exige intervenir un solo archivo y ninguna escena necesita saber cómo se almacenan los datos.

El segundo rasgo de diseño es que el repositorio no conoce la interfaz. Después de cada modificación emite ProgressChanged, una notificación que envía sin saber quién la escucha, y son los componentes de la pantalla de selección los que deciden suscribirse y reaccionar. Esta dirección de la dependencia es deliberada, porque permite añadir o retirar elementos visuales que reaccionen al progreso sin tocar el repositorio, y evita que un componente de persistencia contenga referencias a botones, candados o estrellas.

4.2.4.4 StarRatingCalculator

StarRatingCalculator traduce un número de errores en una calificación de cero a tres estrellas aplicando tres umbrales configurables desde el Inspector. Es un componente de cálculo puro: recibe un valor, devuelve otro y no conserva estado entre llamadas, no accede al almacenamiento y no modifica la interfaz. Esa ausencia de estado es lo que permite invocarlo desde cualquier controlador sin preparación previa y lo que hace que su comportamiento sea completamente predecible a partir de sus parámetros.

Conviene precisar su alcance real dentro de la versión revisada, porque el nombre podría sugerir un uso más extendido del que tiene. Únicamente el Mundo de las Emociones resuelve la calificación mediante este componente. Los mundos Musical y de los Tamaños la calculan dentro de su propio controlador restando los errores a la puntuación máxima, y el Mundo de los Cuentos aplica umbrales de similitud sobre el porcentaje de coincidencia de la lectura. Esta diferencia se conserva porque cada mecánica produce un tipo de dato distinto y forzar un criterio único reduciría un resultado continuo de similitud a un simple recuento de fallos.

Exponer los umbrales como campos serializados en lugar de fijarlos en el código responde a que son precisamente los valores que con mayor probabilidad requerirán ajuste después de una prueba con usuarios reales. Modificar la dificultad de una actividad se resuelve así cambiando tres números desde el editor, sin recompilar el proyecto ni arriesgar una modificación accidental de la lógica de validación.

4.2.4.5 AudioManager

AudioManager centraliza la música de fondo, los efectos puntuales y los volúmenes del videojuego. Está implementado como componente único accesible desde cualquier escena y evita conservar instancias duplicadas, condición necesaria para que la música no se reinicie ni se superponga consigo misma al cambiar de pantalla. Distingue dos fuentes de audio con funciones separadas: una destinada a la música, que puede reproducirse en bucle, y otra reservada a los efectos, que se disparan mediante PlayOneShot sin interrumpir lo que ya está sonando.

Los volúmenes maestros y de música se conservan en el almacenamiento local y se reaplican durante la inicialización del componente, antes de que suene el primer sonido.

Ese orden no es accidental aplicar la configuración después de iniciar la reproducción produciría un salto de volumen perceptible al arrancar la aplicación. El resultado es que la preferencia elegida por el usuario en una sesión anterior se respeta desde el primer fotograma del menú.

Centralizar el audio en un solo componente evita que cada minijuego administre por su cuenta la configuración global. Los controladores de las cuatro actividades solicitan una respuesta sonora concreta y no necesitan saber en qué volumen se encuentra el sistema, si existe música reproduciéndose o cómo se guardan las preferencias. Esta distribución permite además que un cambio en la política de audio, como añadir un control independiente para los efectos, se resuelva en un único archivo.

4.2.4.6 WorldSelectionManager

WorldSelectionManager gobierna la pantalla de selección de mundos y concentra las decisiones que no pertenecen a ningún minijuego concreto: disponibilidad de cada mundo, bloqueo, mensajes de acceso, reinicio del progreso y solicitud de carga de escena. Recibe como entrada el clic sobre una tarjeta y la información que le devuelve WorldProgressRepository, y produce como salida el estado visible de las cuatro tarjetas y, cuando corresponde, la orden de navegación. Su comportamiento en el recorrido completo se describe en el apartado 4.2.7.

El componente no calcula por sí mismo el estado histórico del jugador. Consulta el repositorio y transforma esa información en propiedades visibles, lo que mantiene una frontera clara: una tarjeta no necesita saber cómo se guardan las estrellas y el repositorio no necesita saber cómo se dibuja un candado. Esta separación fue la que permitió que la

pantalla de selección se actualice al regresar de una actividad sin recargar la escena, ya que basta con volver a interpretar los datos que llegan por la notificación de cambio.

La comprobación del desbloqueo se realiza en dos momentos distintos y de forma deliberadamente redundante: al refrescar la interfaz, para decidir el aspecto de la tarjeta, y al recibir el clic, para proteger la navegación. Mantener ambas verificaciones evita que una representación desactualizada o una modificación accidental del botón en el Inspector permita abrir un mundo cuyo requisito todavía no se ha cumplido, situación que no produciría ningún error visible y que resultaría difícil de detectar en una revisión posterior.

4.2.4.7 WorldCardView y WorldStarsPanelView

Estos dos componentes representan visualmente el estado de la pantalla de selección. WorldCardView gobierna el aspecto de una tarjeta de mundo, es decir, si aparece disponible o con candado y cuántas estrellas muestra, mientras que WorldStarsPanelView administra el panel que resume el avance acumulado. Ambos se suscriben a la notificación de cambio del repositorio de progreso al activarse y cancelan esa suscripción al desactivarse, de modo que solo reaccionan mientras la pantalla está efectivamente en uso.

Ninguno de las dos tomas decisiones sobre el juego. No deciden si un mundo puede abrirse, no calculan estrellas y no cargan escenas: reciben el dato ya resuelto y lo traducen en sprites, colores y elementos visibles. Esa restricción es la que permite que aparezcan siempre como destino de una relación y nunca como origen, tal como refleja la lectura de la Tabla 17. Un componente de presentación que además decidiera reglas obligaría a revisar la lógica del juego cada vez que se rediseñara la pantalla.

Las tarjetas constituyen además un caso particular dentro de la interfaz. Su transición automática de color está desactivada porque necesitan combinar dos

informaciones distintas al mismo tiempo, que son el estado de bloqueo y la respuesta al puntero, y esa combinación no puede resolverse con el tinte que Unity aplica por sí solo. Por esa razón su comportamiento visual lo gobierna este componente y no la configuración estándar del botón, decisión que también explica sus anclajes en posiciones fraccionarias descritos en el apartado 4.2.8.

4.2.4.8 WorldSelectionVisuals

WorldSelectionVisuals reúne los efectos visuales de la pantalla de selección que dependen del estado de desbloqueo, de manera que la información sobre el avance no se transmita únicamente mediante un candado. Es el cuarto de los componentes suscritos a la notificación del repositorio de progreso y, como los anteriores, cancela su suscripción al desactivarse para no seguir recibiendo avisos cuando la escena ya ha cambiado.

Se mantuvo separado de WorldCardView porque las dos responsabilidades tienen alcances distintos: la vista de la tarjeta describe un elemento concreto y repetido cuatro veces, mientras que estos efectos actúan sobre el conjunto de la pantalla. Reunirlos en un solo componente habría obligado a que cada tarjeta conociera el estado de las demás para decidir un efecto global, acoplamiento innecesario que se evita dejando que ambos escuchen de forma independiente la misma notificación. Añadir o retirar un efecto de este tipo no requiere, por tanto, modificar ni las tarjetas ni el repositorio.

4.2.4.9 MundoMusicalSequenceGame

MundoMusicalSequenceGame es el controlador de la actividad de secuencias musicales y concentra en un único archivo la temporización de la demostración, el orden esperado de las notas y el estado de aceptación de entrada. Recibe como entrada el clic sobre una de las siete teclas numeradas y el botón de escucha, y produce la iluminación de

las teclas, el sonido correspondiente, la retroalimentación de acierto o error y, al cerrarse la actividad, la calificación que entrega al repositorio de progreso. El flujo completo de la mecánica y su máquina de estados se documentan en el apartado 4.2.6 y en la Ilustración 37.

La razón de que estas tres responsabilidades no se repartieran entre varios archivos es que forman parte de la misma máquina de flujo. El controlador debe bloquear la entrada mientras reproduce la secuencia y volver a habilitarla justo al terminar, y esa decisión depende simultáneamente del momento de la reproducción y del punto de la secuencia en que se encuentra la comparación. Separar la temporización de la validación habría obligado a mantener sincronizados dos componentes que en realidad describen un solo estado, con el riesgo de aceptar pulsaciones durante la demostración.

Las secuencias no están escritas dentro del código sino configuradas como datos serializados, donde cada PianoSequence contiene los pasos que señalan la tecla correspondiente. El controlador recorre esas listas sin conocer de antemano su longitud, de modo que modificar el orden o la cantidad de notas se resuelve desde el Inspector sin reescribir la lógica de comparación. Esta decisión separa el ajuste de dificultad del algoritmo que valida la respuesta, que es precisamente la parte que no debería cambiar entre versiones.

4.2.4.10 MusicalKeyVisual

MusicalKeyVisual administra el aspecto de una tecla del piano: color, brillo y transición entre estados. Cada una de las siete teclas incorpora una instancia de este componente, que recibe del controlador musical la indicación de qué debe representar en cada momento, sea la demostración de la secuencia, un acierto, un error o el bloqueo de la

entrada. El componente no interpreta esa información ni decide por sí mismo cuándo corresponde cada estado.

La separación entre la comparación de la secuencia y la presentación de la tecla evita mezclar dos temporizaciones que responden a criterios distintos. La duración de una animación de pulsación se elige por razones de legibilidad visual, mientras que la pausa entre notas durante la demostración forma parte de la dificultad de la actividad. Al mantenerlas en componentes diferentes, ajustar el estilo de las teclas no altera el ritmo de la secuencia y modificar la dificultad no obliga a rehacer la animación.

Esta organización explica además por qué las teclas tienen desactivada la transición automática de color que Unity aplica a los botones. Un botón estándar solo distingue los estados normal, resaltado, presionado, seleccionado y deshabilitado, y esa lista no contempla la diferencia entre una tecla iluminada por la demostración y una tecla pulsada correctamente por el usuario. Al gobernar el aspecto desde este componente, esa distinción queda disponible sin recurrir a estados artificiales del botón.

4.2.4.11 MusicalTutorialController

MusicalTutorialController presenta las instrucciones previas a la actividad musical y conserva la marca que indica si el tutorial ya se completó. Su entrada son los botones de avance y retroceso entre los pasos de la explicación y su salida es un evento de finalización que el controlador de la actividad espera antes de habilitar la entrada. La marca se guarda en el almacenamiento local mediante una clave propia de este mundo, de manera que las instrucciones aparezcan únicamente en la primera visita y no interrumpan los intentos posteriores.

El tutorial se resolvió como una capa dentro de la misma escena y no como una escena independiente. Esa decisión evita una carga intermedia entre la explicación y la actividad, que resultaría especialmente perceptible en un recorrido que el usuario puede repetir varias veces. Como contrapartida, todos los elementos del tutorial existen en memoria desde el arranque de la escena, coste asumible dado que se trata de paneles de interfaz y texto.

La capa del tutorial se deja desactivada en la escena de forma deliberada y solo se activa cuando el controlador comprueba que corresponde mostrarla. Si se dejara activa y se ocultara por código durante la inicialización, aparecería visible durante una fracción de segundo mientras la escena termina de cargar. Este mismo criterio se aplica en los demás mundos que disponen de tutorial y forma parte de las comprobaciones al crear una escena nueva, según se detalla en el apartado 4.2.10.

4.2.4.12 VoiceRecognitionTest

VoiceRecognitionTest es el controlador del Mundo de los Cuentos y coordina cuatro etapas sucesivas: la selección del relato, la preparación del micrófono, la escucha y la evaluación del resultado. No realiza por sí mismo el reconocimiento ni la comparación lingüística; delega la captura en el servicio de voz y la valoración del texto en ReadingEvaluator, de modo que su responsabilidad se limita a decidir en qué etapa se encuentra la actividad y qué debe mostrar la interfaz en cada una. El detalle de la comprobación previa del micrófono y de los filtros aplicados al texto entrante se documenta en el apartado 4.2.6.

La razón de que la captura y la evaluación se mantengan fuera de este controlador es que la entrada de esta actividad depende de una tecnología ajena al resto del videojuego.

El micrófono y el servicio de dictado pueden fallar, detenerse o entregar resultados parciales sin que exista ningún error en la lógica del cuento, y mezclar esas dos fuentes de fallo en un solo archivo haría muy difícil determinar dónde se produjo un problema. Con la distribución adoptada, la escucha, la transcripción y la valoración del texto pueden comprobarse por separado.

El controlador cancela su suscripción a los eventos del servicio de voz al destruirse, y esa cancelación no es un detalle accesorio. El servicio de dictado del sistema operativo puede continuar activo durante el cambio de escena y, sin liberar la suscripción, intentaría notificar a un componente que ya no existe. Esa es también la razón por la que el controlador detiene sus corrutinas y libera el micrófono al desactivarse, en lugar de confiar en que la carga de la escena siguiente lo resuelva.

4.2.4.13 ISpeechToTextService

ISpeechToTextService no es un componente ejecutable sino un contrato: declara qué operaciones debe ofrecer cualquier motor de reconocimiento de voz que quiera usarse en el proyecto. Ese contrato comprende las operaciones de iniciar y detener la escucha junto con los eventos de resultado parcial, resultado final, error y cambio de estado. El controlador del cuento trabaja exclusivamente contra esta declaración y nunca contra un motor concreto.

La utilidad de un contrato solo se aprecia al considerar el mantenimiento a medio plazo. El reconocimiento de voz es la única parte del videojuego que depende de un servicio del sistema operativo, y es también la que con mayor probabilidad deberá sustituirse si el prototipo se amplía a otras plataformas. Al existir esta separación, cambiar de motor exige escribir una nueva implementación que cumpla el mismo contrato, pero no

obliga a modificar el controlador de la actividad, la selección de cuentos, la evaluación de la lectura ni la interfaz de usuario.

La declaración de eventos diferenciados para resultado parcial y resultado final responde al comportamiento real de los motores de dictado, que entregan hipótesis provisionales mientras el usuario todavía habla y confirman el texto una vez concluida la frase. Recoger ambos casos en el contrato permite que la interfaz muestre lo que se está reconociendo en tiempo real y que la evaluación se aplique solo sobre el texto ya confirmado.

4.2.4.14 WindowsDictationSpeechService

WindowsDictationSpeechService es la implementación concreta del contrato anterior para el entorno Windows. Encapsula DictationRecognizer, que es el componente de reconocimiento que ofrece el sistema operativo, y traduce sus eventos a los que declara ISpeechToTextService. Este archivo es el único punto del proyecto que conoce la interfaz de programación específica de Windows, y esa concentración es intencionada.

Reunir la dependencia del sistema operativo en un solo archivo tiene una consecuencia documental además de técnica: permite delimitar con precisión qué parte del prototipo no puede generalizarse fuera del entorno comprobado. El módulo de voz depende de Windows, del micrófono y de sus permisos, de modo que cualquier afirmación sobre su funcionamiento queda acotada a la configuración descrita en el apartado 4.2.11, mientras que el resto de la actividad de lectura no está sujeta a esa limitación.

El servicio también absorbe las condiciones de cierre que no proceden de la lógica del cuento, como el silencio prolongado, el error del propio servicio y el tiempo máximo de escucha. Traducirlas a eventos del contrato evita que el controlador tenga que interpretar

códigos de error del sistema operativo y mantiene la actividad legible: lo que el controlador recibe es siempre un resultado, un error o un cambio de estado, con independencia de la causa técnica que lo haya originado.

4.2.4.15 ReadingEvaluator

ReadingEvaluator transforma dos textos, el esperado y el reconocido, en un resultado cuantificable. Realiza la normalización de ambos, cuenta las coincidencias y aplica los umbrales que convierten el porcentaje de similitud en estrellas. Es el único componente que decide si una lectura se considera correcta, y lo hace sin tener ninguna relación con el micrófono, con la interfaz ni con el progreso. El procedimiento de normalización, incluido el tratamiento especial de la letra ñ, se detalla en el apartado 4.2.6.

Mantener la evaluación fuera del controlador principal responde a una razón práctica de comprobación. Al recibir texto en lugar de audio, este componente puede verificarse de forma determinista sin necesidad de iniciar el micrófono ni de reproducir condiciones acústicas concretas, lo que permite separar los errores de procesamiento textual de las condiciones variables del hardware y del servicio de dictado. Esa posibilidad se aprovecha efectivamente en la batería de pruebas descrita en el apartado 4.2.11.

La separación tiene además un efecto sobre la estabilidad del criterio de evaluación. Al no depender de la interfaz de lectura, un cambio en la presentación del cuento, en los botones o en los mensajes mostrados no puede alterar el cálculo de similitud. Esta condición resulta relevante porque el criterio de valoración es lo que debe permanecer constante si en el futuro se repite la comprobación del módulo de voz con otro dispositivo o en otro equipo.

4.2.4.16 StoryProgressRepository

StoryProgressRepository conserva el avance del Mundo de los Cuentos con un nivel de detalle que los demás mundos no requieren. Mientras el repositorio general registra un único resultado por mundo, este componente guarda por separado el estado de cada relato, es decir, si se completó, su mejor puntaje y sus mejores estrellas, y mantiene además la marca del tutorial de este mundo. Cuando se cumple la condición de cuentos completados, comunica el avance general a WorldProgressRepository, que es el que sigue gobernando el desbloqueo del mundo siguiente.

Este esquema exige un elemento adicional que conviene explicar porque su necesidad no es evidente: junto a las claves de cada relato se conserva un listado de los identificadores ya registrados. Sin ese listado no sería posible localizar todas las claves individuales al reiniciar el progreso, ya que el número de cuentos registrados no se conoce de antemano y el almacenamiento local no permite recorrer las claves existentes. El listado actúa, por tanto, como el índice que hace posible un borrado completo.

La existencia de un repositorio propio para este mundo responde a que la actividad de lectura no produce el mismo tipo de resultado que las demás. Una secuencia musical o una ronda de emociones se resuelven en un solo intento con un número de errores asociado, mientras que la lectura se compone de varios relatos independientes que el usuario puede abordar en distinto orden y repetir por separado. Registrar solo el mejor resultado global habría impedido conservar el detalle de cada cuento.

4.2.4.17 SizeWorldController

SizeWorldController construye cada ronda del Mundo de los Tamaños a partir de un hábitat, elige el par de animales, decide en qué lado aparece cada uno, formula la consigna

y valida la selección del usuario. Recibe como entrada el clic sobre uno de los dos animales y produce la resolución de la ronda junto con los eventos que permiten a la capa de presentación reaccionar. El filtrado previo de los hábitats y el criterio de validación se describen en el apartado 4.2.6.

La lógica y la presentación se mantienen separadas mediante eventos. Este controlador decide el hábitat, el par de animales, la consigna y la validez de la respuesta, mientras que SizeWorldPresentationController escucha el inicio de ronda, la validación y la finalización para ejecutar las entradas de los animales, las reacciones del personaje guía y la sincronización visual. La división se adoptó para que modificar una animación, una posición de entrada o una pose no pueda alterar la regla que determina cuál de los dos animales es el grande.

Los animales y los hábitats se exponen como estructuras serializadas configurables desde el Inspector, y el controlador descarta las configuraciones incompletas antes de construir una ronda. Esa comprobación previa evita el fallo más probable al ampliar el contenido, que es que un recurso ausente convierta silenciosamente una pregunta en un planteamiento sin respuesta válida. Incorporar nuevos pares es, por tanto, una tarea de configuración siempre que se conserve la clasificación de tamaño que utiliza la mecánica.

4.2.4.18 SizeWorldTutorialController

SizeWorldTutorialController presenta las instrucciones previas al Mundo de los Tamaños y sigue el mismo esquema que el tutorial de la actividad musical. Consulta la marca guardada para este mundo, muestra los pasos de la explicación cuando corresponde y comunica su finalización mediante un evento que el controlador de la actividad espera antes

de habilitar la entrada del usuario. Mientras el tutorial permanece activo, la actividad queda en estado de reposo y no acepta selecciones.

Mantener un controlador de tutorial por cada mundo, en lugar de un componente único reutilizado, responde a que las instrucciones no son intercambiables: el número de pasos, los recursos mostrados y la consigna dependen de la mecánica concreta que se explica. Un componente común habría exigido parametrizar todas esas diferencias y habría trasladado a la configuración una complejidad que en esta escala resulta más clara resuelta por separado. La marca de finalización, en cambio, sí sigue un formato común y se conserva junto al resto de claves de progreso descritas en el apartado 4.2.9.

4.2.4.19 EmotionGameManager

EmotionGameManager gobierna las rondas del Mundo de las Emociones. Presenta una expresión por ronda, espera la selección de un icono, compara ambos valores, lleva el recuento de las ocho rondas previstas y decide cuándo mostrar el resultado. Es el único de los cuatro controladores de mundo que resuelve la calificación mediante StarRatingCalculator en lugar de calcularla internamente. El tratamiento de la emoción opcional de miedo y la regla de no repetir la expresión anterior se documentan en el apartado 4.2.6.

La decisión de diseño más relevante de esta actividad es que el contador de rondas solo avanza después de una respuesta correcta. Un error muestra la retroalimentación, espera un momento y vuelve a habilitar las respuestas sobre la misma expresión, de modo que la duración de la actividad es predecible y lo único que varía entre partidas es el número de intentos acumulados. Esta regla sustituye a un sistema de vidas, que habría podido terminar la actividad antes de que el usuario viera todas las expresiones previstas.

El gestor no contiene la descripción de las emociones ni la lógica de los botones. La vista de ronda describe una emoción y los recursos que la representan, y cada botón conoce únicamente el valor que le corresponde. Al conservar aquí solo la comparación y el recuento, añadir o retirar una emoción se reduce a configurar sus recursos y su valor asociado, sin duplicar reglas de validación en cada opción de respuesta.

4.2.4.20 EmotionAnswerButton y EmotionResultPanel

Estos dos componentes cubren los extremos de la interacción en el Mundo de las Emociones. EmotionAnswerButton identifica la emoción que representa y transmite esa selección al gestor cuando el usuario pulsa; no conoce el contador de rondas, ni las estrellas, ni el avance de la actividad. EmotionResultPanel presenta el cierre, es decir, la calificación obtenida y las opciones de repetir o regresar al selector, una vez que el gestor ha resuelto las ocho rondas.

Situar la comparación en el gestor y no dentro de cada botón evita que la regla de validación quede repetida tantas veces como opciones existan. Si cada botón decidiera por sí mismo si la respuesta es correcta, añadir una emoción implicaría duplicar esa decisión y cualquier cambio en el criterio obligaría a revisar todos los botones, con el riesgo de que uno quedara desactualizado. Con la distribución adoptada existe una única ruta de validación y de cierre de la actividad.

La coordinación entre ambos componentes explica también el comportamiento de la emoción opcional. Cuando esa emoción no se incluye, el gestor oculta su botón de respuesta, de manera que las opciones visibles y las expresiones posibles permanecen sincronizadas. Sin esa relación, el usuario podría encontrar un botón que nunca corresponde

a la respuesta correcta, situación que no produciría ningún error técnico pero sí una actividad incoherente.

4.2.4.21 CharacterDialogueController

CharacterDialogueController administra el personaje guía: sus diálogos, sus poses y la retroalimentación que acompaña a las acciones del usuario. Recibe la indicación de qué debe expresar y se ocupa de mostrar el texto, cambiar la imagen del personaje y sincronizar la aparición del mensaje. No consulta el progreso, no valida respuestas y no conoce las reglas de ninguna de las cuatro actividades.

La forma en que este componente cambia de expresión está determinada por el pipeline artístico descrito en el primer bloque del capítulo. Las poses se exportaron como archivos PNG independientes precisamente porque los componentes visuales de Unity trabajan con referencias concretas a un sprite, de modo que el controlador no dibuja ni modifica la imagen: sustituye una referencia por otra pose ya preparada. Esta decisión mantiene el arte fuera de la lógica y permite corregir una pose en la herramienta de edición y reemplazarla sin tocar el código.

El componente tampoco decide cuándo debe reaccionar. Esa decisión llega desde el puente descrito en el apartado siguiente, lo que mantiene al personaje completamente desvinculado de las mecánicas. Gracias a esa distancia, un cambio en el texto, en la pose o en el ritmo de aparición del mensaje no afecta a los algoritmos que validan la música, la lectura, los tamaños o las emociones.

4.2.4.22 GameplayDialogueFeedbackBridge

GameplayDialogueFeedbackBridge conecta los cuatro mundos con el personaje guía sin que unos y otro se conozcan. Se suscribe a los eventos de validación que publican

los cuatro controladores y traduce cada resultado en la reacción correspondiente, sea un mensaje de felicitación o una invitación a intentarlo de nuevo. Es el único archivo del proyecto que sabe simultáneamente qué ha ocurrido en una mecánica y qué debe expresar el personaje.

Este componente existe porque la alternativa habría sido que cada controlador guardara una referencia al personaje y lo invocara directamente. Esa solución habría funcionado, pero habría introducido en las cuatro mecánicas una dependencia de un elemento narrativo que no tiene relación con la validación de una respuesta, y habría obligado a modificar los cuatro controladores cada vez que cambiara la forma de dar retroalimentación. Con el puente, los controladores se limitan a publicar lo ocurrido y desconocen por completo que exista un personaje.

La consecuencia práctica se aprecia al mantener el proyecto. Retirar el personaje guía de una actividad, cambiar sus mensajes o añadir una reacción nueva son operaciones que se resuelven en este archivo y en el controlador de diálogo, sin abrir ninguno de los controladores de mundo. Del mismo modo, una actividad nueva solo necesita publicar sus eventos de validación para quedar integrada en el sistema de retroalimentación, tal como indica el procedimiento del apartado 4.2.15.

4.2.4.23 UIStarDisplay, UIPanelTransition y UIProgressBar

Estos tres componentes resuelven elementos de interfaz que se repiten en varias escenas y que, de no existir, tendrían que reescribirse en cada una. UIStarDisplay representa la calificación obtenida mediante estrellas y anima su aparición; UIPanelTransition ejecuta el cambio entre los paneles de inicio, juego y resultado dentro de

una misma escena; y UIProgressBar muestra el avance de una actividad en curso. Los tres reciben un valor ya calculado y se limitan a representarlo.

Su condición de componentes reutilizables explica por qué no contienen ninguna regla. Una barra de progreso que decidiera cuándo avanzar dejaría de ser reutilizable en el momento en que dos actividades midieran el avance de forma distinta, y una representación de estrellas que calculara la calificación duplicaría un criterio que ya reside en los controladores. Al recibir el dato resuelto, los mismos componentes sirven a las cuatro actividades pese a que cada una calcule su resultado con una regla propia.

UIPanelTransition responde además a una decisión de organización descrita en el apartado 4.2.8: las actividades no cargan una escena distinta para cada fase, sino que activan y desactivan paneles dentro de la misma escena. Ese esquema evita recargas intermedias y hace inmediato el paso de la consigna al juego y del juego al resultado, a costa de que los tres paneles existan en memoria desde el arranque, coste asumible dado el tamaño de las escenas del proyecto.

4.2.4.24 MicrophoneSettings y AudioOptionsController

Ambos componentes pertenecen a la pantalla de opciones y administran la configuración que el usuario puede modificar. MicrophoneSettings permite seleccionar el dispositivo de entrada y probarlo antes de utilizarlo en la actividad de lectura, conservando el nombre del dispositivo elegido en el almacenamiento local mediante una clave propia. AudioOptionsController gobierna los volúmenes maestros y de música, que entrega a AudioManager y que se conservan igualmente entre sesiones.

La prueba de micrófono se incorporó porque el fallo más frecuente en una actividad basada en voz no es un error de programación sino una condición del equipo: un dispositivo

no seleccionado, un permiso denegado o un micrófono silenciado. Disponer de una comprobación previa permite distinguir esos casos antes de iniciar la actividad, en lugar de que el usuario interprete la ausencia de resultado como un defecto del videojuego. Esta comprobación es independiente de la que el propio controlador del cuento realiza al comenzar cada lectura.

Conviene registrar que la selección del dispositivo presenta una limitación en Windows, tal como recoge la Tabla 20, y que por ello la verificación del módulo de voz debe realizarse en el entorno de entrega documentando el idioma del sistema, el dispositivo utilizado y las condiciones básicas de escucha. Esa limitación no afecta al resto del videojuego, ya que ninguna otra actividad depende de la entrada de audio.

4.2.4.25 ManagerVideo

ManagerVideo administra la configuración de vídeo de la aplicación: resolución, modo de pantalla, sincronización vertical y límite de fotogramas. Constituye la única excepción al mecanismo general de persistencia del proyecto, ya que no guarda sus valores en el sistema de preferencias sino en un archivo JSON alojado en el directorio de datos persistentes de la aplicación, según se registra en la Tabla 20.

Esa excepción responde a la naturaleza del dato. Mientras el progreso se compone de valores independientes entre sí, la configuración de vídeo agrupa cuatro parámetros que solo tienen sentido en conjunto: una resolución sin su modo de pantalla, o una sincronización vertical sin su límite de fotogramas, describen un estado incompleto. Tratarlos como un objeto único evita que una escritura interrumpida deje la configuración a medio aplicar y facilita restaurarla o descartarla en bloque.

El componente opera dentro del marco fijado por la configuración de compilación, que establece una resolución base de 1024 por 768 píxeles, arranque en pantalla completa y ventana no redimensionable. Esa última condición es coherente con el escalado de interfaz descrito en el apartado 4.2.8, porque descarta las proporciones arbitrarias que produciría una ventana ajustable libremente por el usuario y mantiene previsible el comportamiento de los anclajes.

4.2.4.26 WorldSelectionSceneBuilder y SizeWorldSceneBuilder

Estos dos archivos constituyen un caso distinto del resto de la tabla, porque no forman parte del videojuego. Son herramientas que se ejecutan únicamente dentro del editor de Unity, se invocan desde su menú y su función es generar la estructura de una escena con los elementos que el proyecto exige, evitando construirla manualmente objeto por objeto. Quedan excluidos del ejecutable, de modo que su presencia no incrementa el tamaño de la compilación ni interviene en el funcionamiento de la aplicación entregada.

Su utilidad se comprende al revisar cuántos elementos debe contener una escena para funcionar correctamente. Toda escena necesita un lienzo configurado en superposición con la misma resolución de referencia, un sistema de eventos con el módulo de entrada, la capa de transición visual y los tres paneles de inicio, juego y resultado. Omitir cualquiera de ellos no impide compilar, pero produce fallos que solo se manifiestan al ejecutar y que la batería de pruebas detecta como componentes ausentes.

Automatizar esa construcción reduce el error humano en la parte más repetitiva del trabajo y garantiza que las escenas nuevas partan de la misma estructura que las existentes. Por esa razón el procedimiento de ampliación descrito en el apartado 4.2.15 sitúa la

creación de la escena inmediatamente después de la preparación de los recursos gráficos y antes de escribir el controlador de la actividad.

El recorrido por los componentes anteriores permite reconocer un criterio repetido a lo largo de todo el proyecto: cada archivo conserva una única responsabilidad y las dependencias apuntan siempre en la misma dirección, desde la presentación hacia la lógica y desde la lógica hacia los servicios compartidos, sin que ninguna capa inferior necesite conocer a la superior. Los repositorios ignoran la interfaz, los componentes visuales ignoran las reglas y el servicio de voz ignora el contenido de los cuentos. Esa disciplina es la que hace posible que las modificaciones descritas en los apartados 4.2.14 y 4.2.15 queden acotadas a un número reducido de archivos y que la comprobación de extremo a extremo documentada en el apartado 4.2.11 pueda interpretarse como una verificación del conjunto y no de una sola parte.

4.2.5 Desarrollo de sistemas principales

Las decisiones de implementación responden a cuatro criterios: navegación visible, actividades breves, retroalimentación inmediata y separación modular entre escenas, controladores, servicios y recursos. La solución se ejecuta de forma local, almacena el progreso sin base de datos y limita el reconocimiento de voz al servicio disponible en Windows.

El alcance documental distingue los componentes implementados, las evidencias de funcionamiento y las relaciones comprobadas entre escenas, controladores y servicios. El apartado 4.2.11 reúne la verificación técnica, mientras que los apartados anteriores explican cómo cada sistema recibe la entrada, procesa sus reglas, actualiza el estado y entrega el resultado visible dentro del videojuego.

Los requerimientos funcionales y no funcionales establecidos para el prototipo se describen a continuación y orientan la implementación de los sistemas principales.

La solución debe abrir el menú principal, presentar la selección de mundos, ejecutar una actividad por cada temática, validar la respuesta del usuario y mostrar retroalimentación. Además, debe registrar estrellas y avance, conservar el progreso entre sesiones, permitir el retorno al selector e iniciar la lectura apoyada por voz cuando el sistema disponga de micrófono y permisos.

El prototipo debe ejecutarse en PC Windows de 64 bits, mantener legibilidad en las resoluciones documentadas y conservar los archivos de la carpeta de compilación. La interfaz debe responder al mouse y ofrecer mensajes comprensibles. La validación final requiere una compilación sin referencias rotas, pruebas reproducibles y una identificación inequívoca de versión, fecha y configuración.

A partir de esos requerimientos se construyeron cuatro sistemas transversales que sostienen el funcionamiento de las cuatro actividades: navegación, progreso, estrellas y audio. Cada uno se describe a continuación indicando el problema que resuelve, dónde reside, qué lo pone en marcha, qué datos lee o escribe y qué produce en la interfaz.

MundoAprendoSceneNames y SceneNavigation resuelven el traslado entre escenas y previenen un fallo frecuente: intentar cargar una escena que no está registrada en la configuración de compilación. El primer script declara los nombres de las seis escenas como constantes, de modo que ningún otro archivo necesita escribir un nombre de escena de forma literal. SceneNavigation recibe la solicitud de carga, comprueba que el destino sea válido y delega la transición visual en UISceneTransition antes de abrir la escena. Esta

separación permite modificar el efecto de transición sin mezclarlo con la decisión de qué pantalla debe cargarse.

WorldProgressRepository conserva entre sesiones qué mundos se completaron y con cuántas estrellas. Está implementado como una clase estática, por lo que no requiere un GameObject propio en cada escena. Recibe como entrada un índice de mundo, comprendido entre cero y tres, y un número de estrellas. La escritura la solicita el controlador del mundo al terminar la actividad, y el repositorio conserva el mejor resultado histórico, de manera que repetir una actividad con peor desempeño no reduce el avance ya obtenido. Después de cada modificación emite ProgressChanged, evento que permite a la pantalla de selección actualizar bloqueos y estrellas sin acoplarse al modo en que PlayerPrefs almacena los datos.

StarRatingCalculator concentra el cálculo común de estrellas a partir del número de errores y de tres umbrales configurables. Conviene precisar su alcance real: en la versión revisada lo utiliza el Mundo de las Emociones. Los mundos Musical y de los Tamaños resuelven el cálculo dentro de su propio controlador restando los errores a la puntuación máxima, mientras que el Mundo de los Cuentos aplica umbrales de similitud sobre la lectura. Esta diferencia se mantiene porque cada mecánica produce un tipo de resultado distinto y evita forzar un criterio único que no represente correctamente la entrada evaluada.

AudioManager centraliza la música de fondo y los efectos. Está implementado como un componente único accesible desde las escenas y evita que se conserven instancias duplicadas. Distingue dos AudioSource: uno para la música, que puede reproducirse en bucle, y otro para efectos puntuales mediante PlayOneShot. Los volúmenes maestro y de

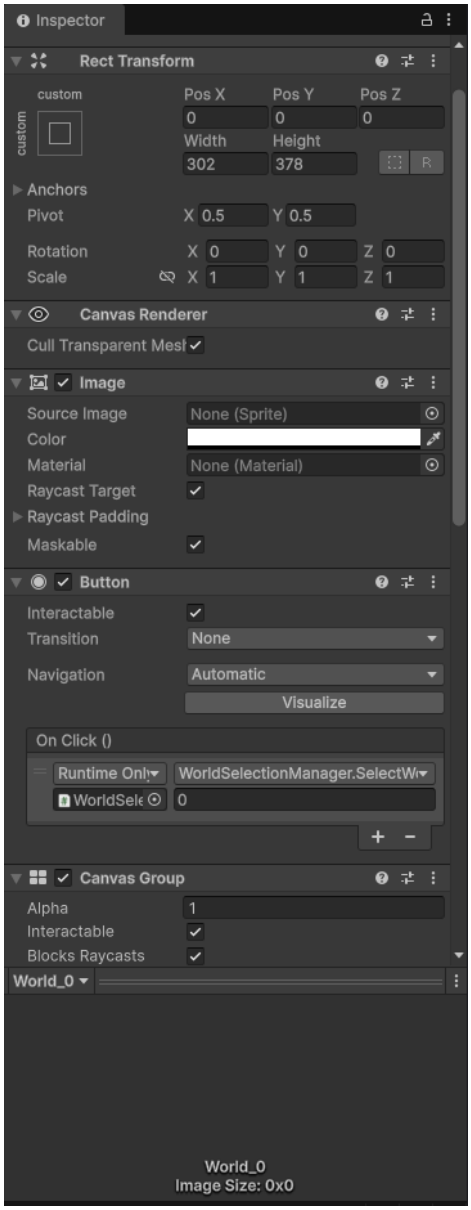
música se conservan en PlayerPrefs y se reaplican al inicializar el componente. Centralizar esta responsabilidad permite que los controladores de los minijuegos soliciten una respuesta sonora sin administrar por separado la configuración global de audio.

Los cuatro sistemas se encadenan en una secuencia única y verificable. La navegación traslada al usuario hasta la escena de una actividad; el controlador de esa actividad valida la entrada y solicita la reproducción de un efecto al sistema de audio; al concluir, aplica la regla de estrellas y entrega el resultado al repositorio de progreso; el repositorio conserva el mejor valor y emite su notificación de cambio; y los componentes de la pantalla de selección, suscritos a esa notificación, actualizan los bloqueos y las estrellas visibles. Comprobar esa cadena completa equivale a comprobar el funcionamiento conjunto del prototipo.

Los eventos de interfaz dirigen la entrada hacia el controlador de la escena; este actualiza la respuesta y solicita el guardado cuando corresponde. Antes de modificar un botón se debe revisar el método enlazado en On Click y las referencias del Inspector. La Ilustración 33 muestra esa conexión en la navegación y en la representación de estrellas.

Ilustración 33

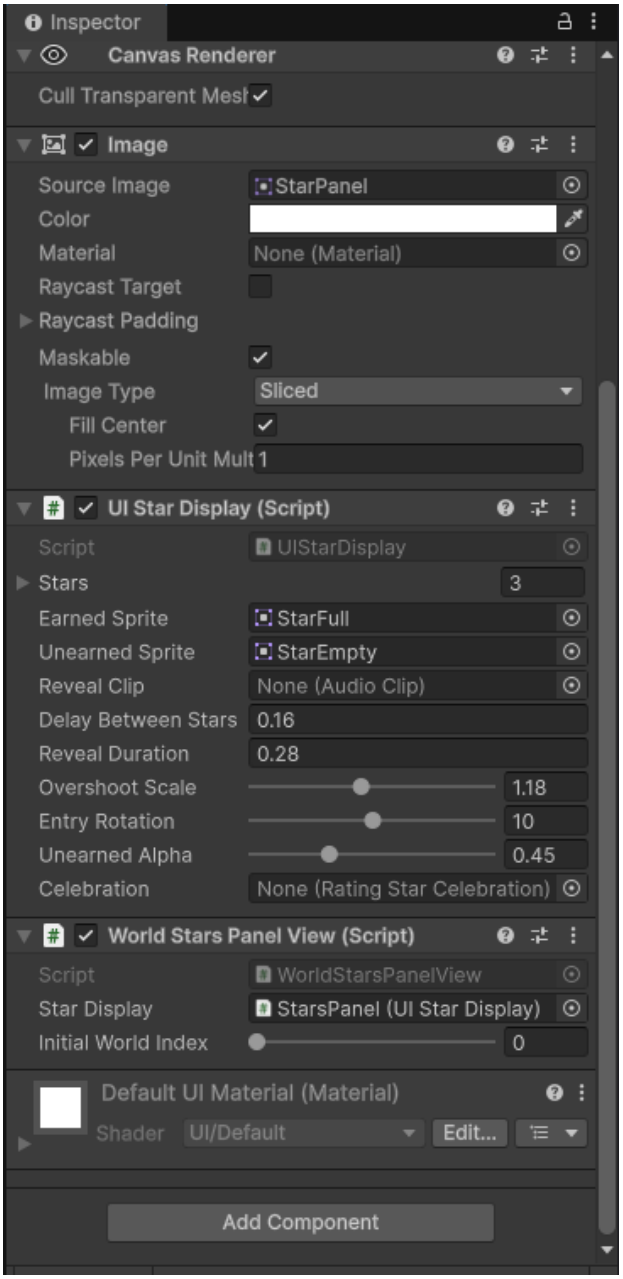
Configuración del evento On Click en la navegación.



Nota. Captura propia del proyecto y de sus registros técnicos.

Ilustración 34

Visualización y configuración de estrellas en la interfaz.



Nota. Captura propia del proyecto y de sus registros técnicos.

4.2.6 Desarrollo de mecánicas del videojuego

Las cuatro actividades emplean el mismo flujo general: reciben una entrada, la comparan con la consigna, muestran retroalimentación y actualizan el resultado. Los

apartados siguientes documentan solo la lógica particular que debe conservarse al modificar cada controlador.

La descripción que sigue se centra en la lógica implementada. Las pantallas tal como las percibe el usuario final se documentan en el Capítulo III.

WorldSelectionManager consulta el progreso guardado, activa las tarjetas disponibles y representa las estrellas. Al añadir o reordenar un mundo se debe actualizar la regla de desbloqueo y comprobar que el reinicio no deje estados inconsistentes.

La selección de mundos se integró como un controlador independiente porque concentra decisiones que no pertenecen a ningún minijuego: disponibilidad, bloqueo, mensajes de acceso, reinicio del progreso y carga de escena. WorldSelectionManager no calcula por sí mismo el estado histórico; consulta WorldProgressRepository y transforma esa información en propiedades visibles de las tarjetas. Esta distribución evita que una tarjeta tenga que conocer cómo se guardan las estrellas y evita que el repositorio conozca cómo se dibuja un candado.

La comprobación de desbloqueo se realiza tanto al refrescar la interfaz como al recibir el clic. La primera determina el aspecto de la tarjeta y la segunda protege la navegación. Mantener ambas verificaciones evita que una representación desactualizada o una modificación accidental del botón permita abrir un mundo cuyo requisito todavía no se ha cumplido.

MundoMusicalSequenceGame implementa la mecánica de repetición de secuencias. Cada secuencia es una lista ordenada de pasos y cada paso apunta a una de las siete teclas mediante su índice. La actividad comienza cuando el usuario pulsa Escuchar: el controlador bloquea la entrada, recorre la secuencia iluminando y haciendo sonar cada tecla con una

pausa configurable entre notas y, solo al terminar la demostración, habilita los botones. A partir de ese momento compara cada pulsación con el paso esperado; un acierto avanza el índice y un error registra el fallo, muestra retroalimentación y, cuando la opción repeatOnMistake está activa, vuelve a reproducir la secuencia antes de permitir otro intento. La lógica se mantuvo en un controlador único porque la temporización, el orden esperado y el estado de aceptación de entrada forman parte de la misma máquina de flujo de la actividad.

Una particularidad técnica de este mundo es la generación del sonido. Si una tecla no tiene asignado un archivo de audio, el controlador sintetiza por software una onda senoidal a la frecuencia declarada para esa tecla, le aplica una envolvente descendente y conserva en memoria el clip generado para reutilizarlo en las siguientes pulsaciones. Ese mecanismo permite que el piano funcione aunque falten los recursos de audio y explica por qué cada tecla declara tanto un clip opcional como una frecuencia.

Las secuencias musicales se configuraron como datos serializados en lugar de codificarse dentro de una cadena fija de instrucciones. Cada PianoSequence contiene pasos que señalan una tecla concreta, y el controlador puede recorrer esas listas sin conocer de antemano su longitud. Esta decisión permite modificar el orden o la cantidad de notas desde el Inspector sin reescribir la lógica de validación, lo que separa la configuración de dificultad del algoritmo que compara la respuesta.

La presentación de cada tecla se delega en MusicalKeyVisual. El controlador musical decide si una pulsación corresponde a demostración, acierto, error o bloqueo, mientras el componente visual administra colores, brillo y transición del botón. La

separación evita mezclar temporizaciones de animación con la comparación de la secuencia y permite cambiar el estilo de las teclas sin tocar el flujo del minijuego.

La elección de una entrada discreta por clic responde además a la naturaleza de la mecánica musical: el dato que debe validarse es el orden de una secuencia y no la posición espacial de un objeto. Al representar cada nota como un botón numerado, el controlador recibe un índice inequívoco y puede compararlo directamente con el paso esperado. Esta integración reduce estados intermedios de interacción, facilita bloquear temporalmente las teclas durante la demostración y permite que la misma lógica funcione aunque cambien la apariencia, el tamaño o la posición visual del piano. Por esa razón, la mecánica se apoya en eventos de botón y en datos serializados, mientras la animación y el sonido actúan como capas de retroalimentación sobre una regla de comparación que permanece independiente.

VoiceRecognitionTest coordina cuatro etapas del Mundo de los Cuentos: selección del cuento, preparación del micrófono, escucha y evaluación. Antes de activar el reconocimiento ejecuta una cuenta atrás y una comprobación previa del micrófono, en la que mide el nivel real de la señal captada durante algo más de un segundo y lo compara con un umbral mínimo. Si no detecta sonido, cancela la operación y muestra un mensaje, en lugar de iniciar una escucha que no produciría texto. Durante la escucha mantiene el estado de la interfaz y recibe las hipótesis o resultados del servicio de voz, mientras que la comparación lingüística se delega a ReadingEvaluator.

La evaluación de la lectura no compara cadenas literales. El texto esperado y el reconocido se normalizan a minúsculas, se les retiran los signos de puntuación y los acentos y, de forma opcional, se descartan las palabras de uso muy frecuente. La normalización conserva deliberadamente la letra ñ mediante un marcador temporal, porque el

procedimiento estándar de descomposición de caracteres la convertiría en n y alteraría palabras del cuento. Sobre las palabras resultantes se cuenta cuántas del texto esperado aparecen en el reconocido y se obtiene un porcentaje de similitud.

El controlador incorpora además dos filtros que responden al comportamiento del motor de dictado de Windows. El primero descarta las palabras reconocidas que no pertenecen al vocabulario del cuento seleccionado, de modo que el ruido ambiental no incremente artificialmente el texto acumulado. El segundo compara el fragmento entrante con el final del texto ya registrado y descarta la parte coincidente, porque el motor reenvía la frase completa cada vez que amplía una hipótesis y, sin ese filtro, las palabras aparecerían duplicadas en pantalla y en la evaluación.

La integración del reconocimiento de voz se dividió en un contrato y una implementación concreta para que la mecánica no dependa directamente de UnityEngine.Windows.Speech. VoiceRecognitionTest trabaja contra ISpeechToTextService y recibe eventos de hipótesis, resultado final, error y estado. WindowsDictationSpeechService encapsula DictationRecognizer y traduce sus eventos al contrato. Con esta estructura, la lógica de selección del cuento, actualización de interfaz y evaluación de lectura permanece separada de la API específica de Windows.

ReadingEvaluator también se mantiene fuera del controlador principal. Su responsabilidad es transformar dos textos en un resultado cuantificable mediante normalización, conteo de coincidencias y umbrales de estrellas. Separar la evaluación de la captura de voz permite probar o modificar los criterios de comparación sin tener que iniciar el micrófono, y evita que un cambio en la interfaz de lectura altere el cálculo de similitud.

La división entre captura, evaluación y presentación fue especialmente importante en esta actividad porque la entrada depende de una tecnología externa al resto del videojuego. El micrófono y DictationRecognizer pueden fallar, detenerse o entregar hipótesis parciales sin que exista un error en la lógica del cuento. Al encapsular esa dependencia en el servicio de voz, VoiceRecognitionTest puede concentrarse en el estado de la actividad y ReadingEvaluator en el criterio de coincidencia. Así, un cambio futuro de proveedor de reconocimiento requeriría sustituir la implementación del servicio y no reescribir la selección de cuentos, el progreso ni la interfaz. La mecánica se integró de esta manera para aislar la dependencia de Windows y mantener verificables por separado la escucha, la transcripción y la valoración del texto.

SizeWorldController construye cada ronda del Mundo de los Tamaños a partir de un hábitat. Antes de plantear la pregunta filtra los hábitats configurados y conserva únicamente aquellos que disponen al menos de un animal grande y uno pequeño con imagen y nombre asignados, lo que garantiza que siempre exista una respuesta correcta e inequívoca. Elegido el hábitat, selecciona un animal de cada tamaño, decide al azar el lado en que aparece cada uno y formula la consigna preguntando por el mayor o por el menor. La generación y la validación permanecen en este controlador, mientras SizeWorldPresentationController recibe los eventos necesarios para animar la presentación y la retroalimentación sin alterar la regla de la ronda.

Una decisión de diseño relevante es que ambos animales se muestran con la misma escala, con independencia de su tamaño real y del hábitat en que aparecen. La respuesta correcta depende exclusivamente del tipo declarado para cada animal y no de su representación en pantalla, de manera que el tamaño visible no funcione como pista. Un

acierto avanza la ronda tras una pausa; un error registra el fallo, sacude la opción elegida y devuelve el control al usuario para que reintente sin perder la ronda.

La escena de Tamaños separa la lógica de la presentación mediante eventos. SizeWorldController decide el hábitat, el par de animales, la consigna y la validez de la selección; SizeWorldPresentationController escucha el inicio de ronda, la validación y la finalización para ejecutar entradas, reacciones del guía y sincronización visual. Esta división se integró para que modificar una animación, una posición de entrada o una pose del personaje no altere la regla que determina cuál animal es grande o pequeño.

Los datos de animales y hábitats se exponen como estructuras serializadas. Esto permite incorporar nuevos pares o cambiar imágenes desde el Inspector siempre que se conserve la clasificación utilizada por la mecánica. El controlador filtra las configuraciones incompletas antes de construir una ronda, de modo que un asset ausente no se transforme silenciosamente en una pregunta sin respuesta válida.

En esta mecánica la selección directa por clic también simplifica la relación entre interfaz y regla. El usuario no necesita desplazar físicamente el animal para expresar una respuesta; basta identificar cuál de las dos opciones cumple la consigna. Técnicamente, cada opción puede enviar al controlador la referencia o categoría del animal seleccionado y la validación se resuelve contra los datos de la ronda. De esta forma, posiciones, animaciones de entrada y tamaños de los sprites pertenecen a la presentación, mientras la decisión grande/pequeño permanece en los datos. Esta separación permite sustituir un personaje, modificar la composición de la pantalla o cambiar una animación sin alterar la condición que determina el acierto.

El controlador del Mundo de las Emociones presenta una expresión por ronda y espera la selección de un icono. Al preparar cada ronda descarta la emoción mostrada inmediatamente antes, siempre que existan alternativas disponibles, para evitar repeticiones consecutivas. La emoción de miedo es opcional y permanece desactivada de forma predeterminada: cuando no se incluye, el controlador oculta también su botón de respuesta, de modo que las opciones visibles y las emociones posibles se mantienen sincronizadas.

Un error no interrumpe la ronda. El controlador muestra la retroalimentación, espera un momento y vuelve a habilitar las respuestas para que el usuario insista sobre la misma expresión. La ronda solo avanza con un acierto, por lo que el número de rondas completadas es fijo y lo que varía entre partidas es la cantidad de errores acumulados.

EmotionGameManager adopta una estructura similar, pero distribuye las responsabilidades entre EmotionRoundView, EmotionAnswerButton y el gestor principal. La vista describe una emoción y sus recursos; cada botón conoce únicamente la emoción que representa y envía la selección; el gestor compara ambos valores, controla las ocho rondas y decide cuándo mostrar el resultado. Esta separación evita que los botones contengan reglas de progreso y facilita añadir o retirar una opción sin duplicar la lógica de validación.

La duración de la actividad se mantiene predecible porque el contador de rondas solo aumenta después de una respuesta correcta. Los errores modifican la calificación final, pero no consumen una ronda. De este modo, el flujo garantiza que el usuario vea el número previsto de expresiones y que el sistema de estrellas refleje la cantidad de intentos necesarios sin introducir un sistema de vidas que pueda terminar la actividad antes de completar su contenido.

La mecánica de Emociones se integró con botones especializados en lugar de colocar la comparación dentro de cada elemento de interfaz. EmotionAnswerButton identifica la emoción que representa y transmite esa selección al gestor; EmotionGameManager conserva el estado de la ronda y decide si coincide con la expresión presentada. El botón, por tanto, no conoce el contador, las estrellas ni el avance. Esta decisión evita duplicar reglas en cada opción de respuesta y hace que agregar una nueva emoción consista principalmente en configurar sus recursos y su valor asociado, manteniendo una única ruta de validación y cierre de la actividad.

Ninguna actividad emplea un sistema de vidas: el usuario puede reintentar cuantas veces necesite y el error solo influye en la calificación final. Ahora bien, el cálculo de esa calificación no es homogéneo entre las cuatro actividades, y conviene documentarlo con precisión. Los mundos Musical y de los Tamaños obtienen las estrellas restando directamente el número de errores a la puntuación máxima de tres. El Mundo de las Emociones emplea el componente común de cálculo, que aplica tres umbrales configurables de errores máximos. El Mundo de los Cuentos no cuenta errores, sino que convierte el porcentaje de similitud de la lectura en estrellas mediante sus propios umbrales. Unificar las tres reglas en el componente común constituye una mejora pendiente que no alteraría el comportamiento actual, pero reduciría la duplicación y concentraría el ajuste de dificultad en un solo archivo.

Las diferencias entre reglas de calificación responden al tipo de dato que produce cada mecánica. Musical y Tamaños generan errores discretos durante una secuencia de intentos; Emociones utiliza umbrales configurables sobre esos errores; Cuentos obtiene un porcentaje de coincidencia entre palabras. Aunque una futura refactorización podría

unificar la interfaz de cálculo, conservar el criterio específico de lectura evita reducir un resultado continuo de similitud a un simple conteo de fallos.

Tabla 18

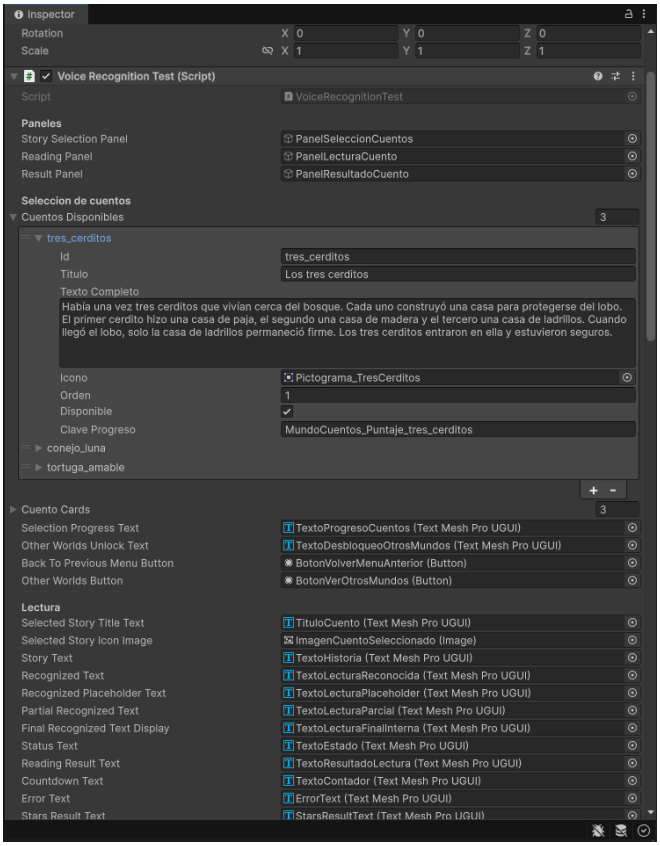
Mecánicas implementadas en los minijuegos.

Actividad	Entrada	Validación implementada	Cierre o continuación
Secuencias musicales	Clic en botones del 1 al 7.	Comparación del orden con la secuencia reproducida.	Nuevo intento o finalización con estrellas.
Lectura con voz	Micrófono y botones de escucha.	Normalización y similitud entre el texto reconocido y el cuento.	Validación, estrellas, repetir o volver.
Comparación de tamaños	Clic sobre uno de dos animales.	Comparación con la respuesta grande o pequeña de la ronda.	Siguiente ronda y resultado final.
Reconocimiento de emociones	Clic en un icono de emoción.	Coincidencia con la expresión mostrada.	Avance entre ocho rondas y resultado.

Nota. Elaboración propia a partir del proyecto y sus registros técnicos.

Ilustración 35

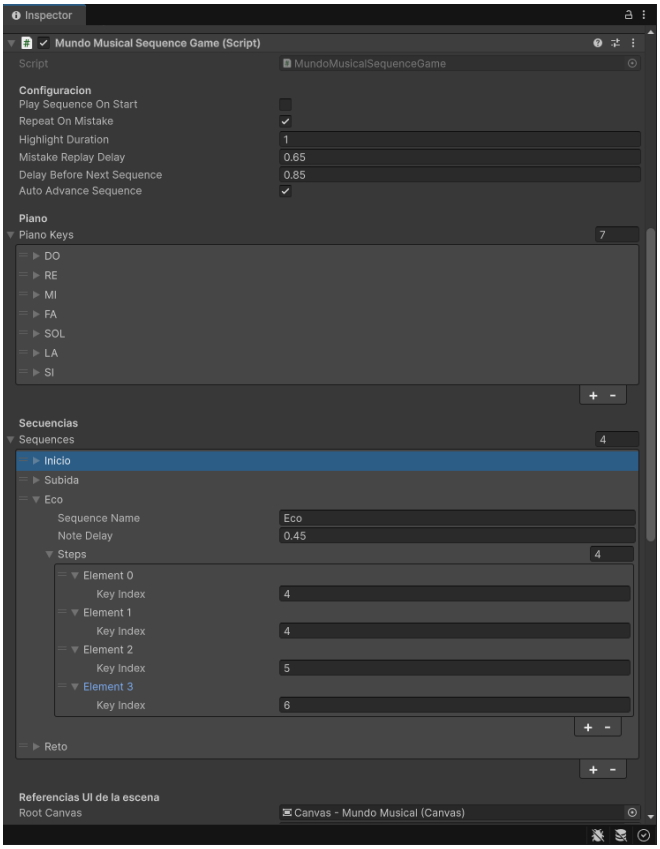
Configuración de la secuencia musical.



Nota. Captura propia del proyecto y de sus registros técnicos.

Ilustración 36

Configuración del controlador de voz.

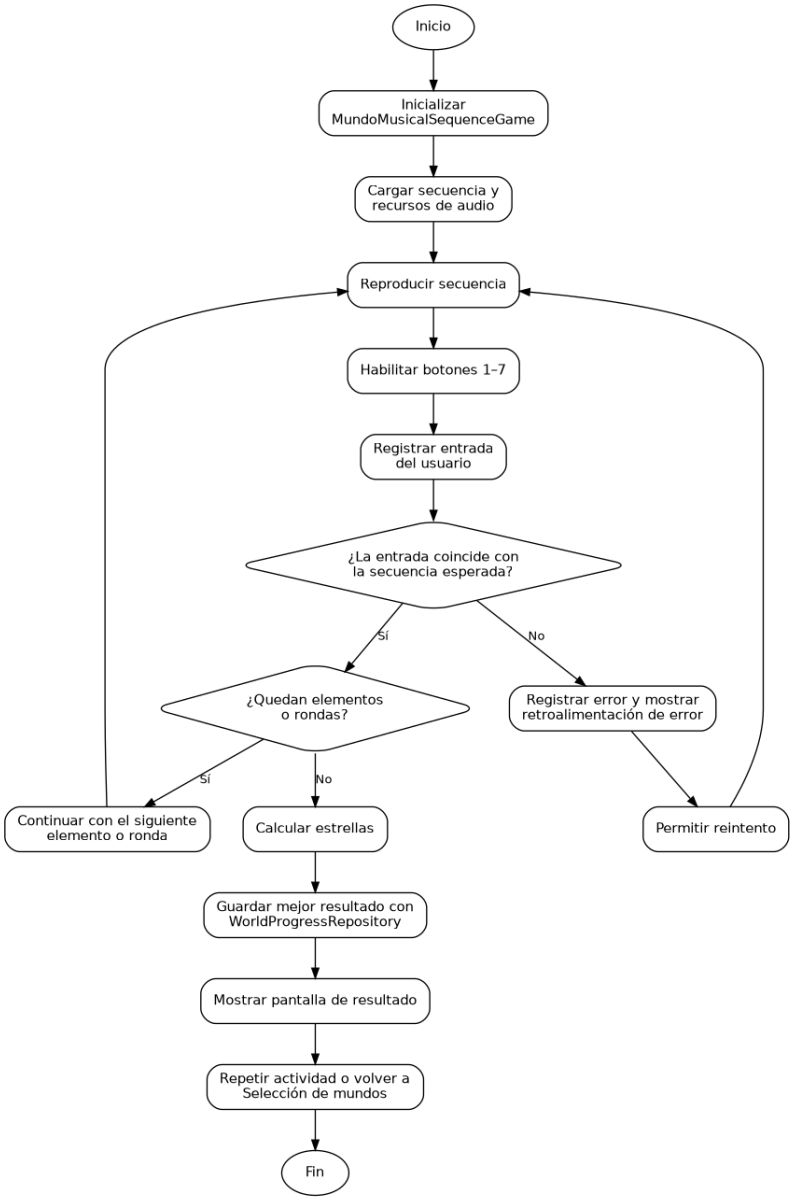


Nota. Captura propia del proyecto y de sus registros técnicos.

La Ilustración 37 representa los estados por los que atraviesa la actividad musical. El recorrido parte de la inicialización del controlador y de la reproducción de la secuencia configurada, continúa con la habilitación de las teclas numeradas y se bifurca en las dos ramas posibles de la comparación. La rama de acierto avanza al paso siguiente o, si la secuencia se ha completado, a la siguiente secuencia; la rama de error devuelve el recorrido al inicio de la secuencia actual. El esquema muestra que el cálculo de estrellas y el guardado ocurren una sola vez, al cerrarse la actividad, y no en cada intento.

Ilustración 37

Estados de ejecución de la actividad de secuencias musicales.



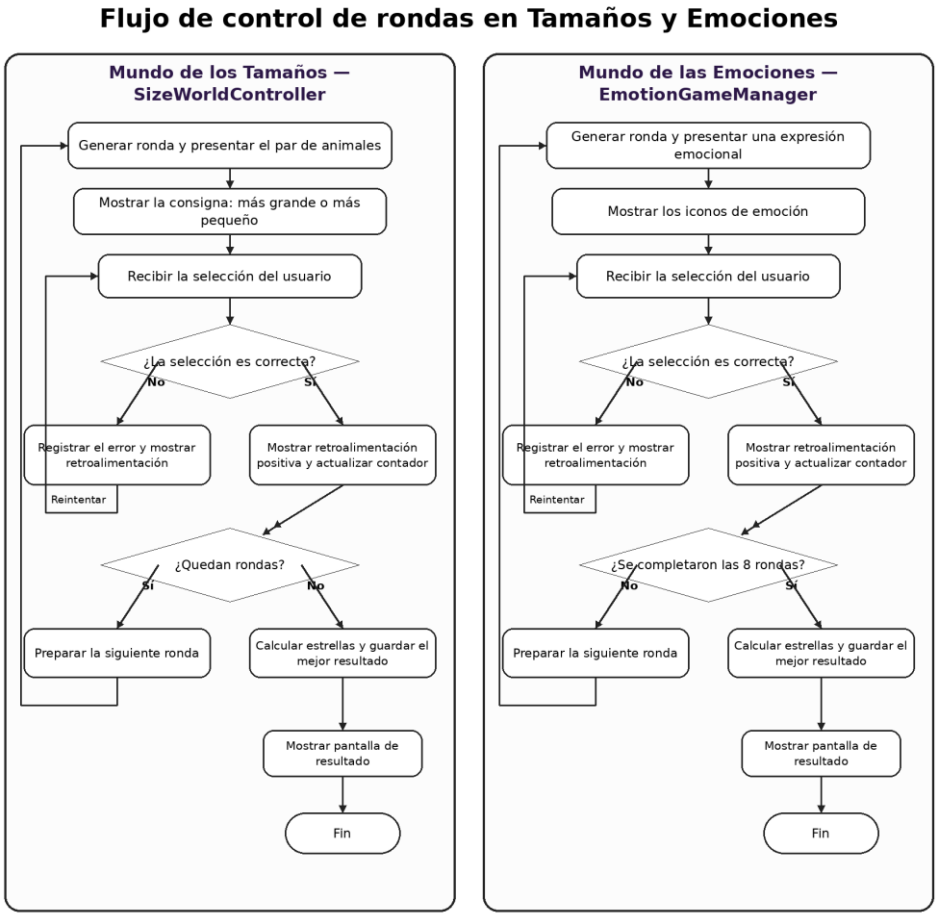
Nota. Elaboración propia a partir de los scripts y de la configuración del proyecto.

La Ilustración 38 presenta los dos recorridos de ronda en paralelo, sin sugerir una herencia que no existe en el código. En el primero, el controlador de Tamaños presenta el par de animales, recibe la selección y valida la relación de tamaño declarada. En el segundo, el controlador de Emociones presenta la expresión, recibe el icono elegido y

administra el recuento de rondas. La comparación de ambos recorridos hace visible su diferencia principal: en Tamaños el error permite reintentar dentro de la misma ronda, mientras que en Emociones la ronda tampoco avanza hasta que se acierta, de modo que en las dos actividades el total de rondas permanece constante y lo variable es el número de errores.

Ilustración 38

Flujo de control de rondas en los mundos de Tamaños y Emociones.



Nota. Elaboración propia a partir de los scripts y de la configuración del proyecto.

4.2.7 Integración de sistemas

Los botones de UGUI reciben el clic izquierdo y llaman métodos de los controladores mediante eventos On Click. Cada revisión debe comprobar los estados

visuales Normal, Highlighted, Pressed, Selected y Disabled, además de los controles específicos para iniciar, detener y validar la lectura por micrófono. La Tabla 19 concentra las interacciones verificadas.

Tabla 19

Controles e interacciones verificadas.

Control	Acción del usuario	Resultado observado
Clic izquierdo	Seleccionar botones, mundos, animales, emociones o números.	La interfaz procesa la opción elegida.
SIGUIENTE / ANTERIOR	Recorrer las instrucciones del tutorial.	Cambia el paso visible del tutorial.
VOLVER	Salir de una actividad o pantalla secundaria.	Retorna al selector o al menú, según la escena.
REINTENTAR / JUGAR OTRA VEZ	Repetir una secuencia o actividad.	Reinicia el intento sin terminar el recorrido.
SELECCIÓN DE MUNDOS	Continuar después de completar una actividad.	Abre SeleccionMundos y actualiza el progreso.
Micrófono	Iniciar, detener y validar la lectura.	Muestra el estado de escucha y procesa el texto reconocido.

Nota. Elaboración propia.

La lógica particular de cada actividad se documenta en el apartado 4.2.6. Este apartado describe el mecanismo que las conecta entre sí y con los servicios comunes. La integración no se resolvió mediante referencias directas entre componentes, sino mediante eventos: cada controlador de mundo expone eventos públicos que anuncian lo ocurrido sin conocer quién los escucha, concretamente la validación de una respuesta, el inicio de una ronda y la finalización de la actividad. Cualquier componente interesado se suscribe a ellos y reacciona por su cuenta.

GameplayDialogueFeedbackBridge se suscribe a los eventos de validación de los cuatro mundos y traduce cada resultado en la reacción correspondiente del personaje guía: un mensaje de felicitación o una invitación a intentarlo de nuevo. Gracias a ese puente, los

controladores de las mecánicas desconocen por completo la existencia del personaje y solo publican el resultado de la acción. Esta decisión desacopla la retroalimentación narrativa de la lógica de juego, por lo que una pose, un texto o una animación del guía puede cambiarse sin modificar los algoritmos que validan música, lectura, tamaños o emociones.

La segunda vía de integración es la notificación de progreso descrita en el apartado 4.2.4. Cuando una actividad concluye y el repositorio registra el resultado, este emite un aviso que recogen los cuatro componentes suscritos de la pantalla de selección. El efecto observable es que, al regresar de un mundo, los bloqueos y las estrellas ya reflejan el estado nuevo sin que la escena haya tenido que recargarse ni consultar el almacenamiento por su cuenta.

El tercer punto de integración es el módulo de voz, que se conecta en sentido inverso al anterior. El servicio de reconocimiento emite eventos de resultado parcial, resultado final, error y cambio de estado, y el controlador del cuento se suscribe a ellos al iniciarse y cancela la suscripción al destruirse. Esa cancelación no es un detalle accesorio: el servicio de dictado del sistema operativo puede seguir activo durante el cambio de escena y, sin liberar la suscripción, intentaría notificar a un componente ya destruido.

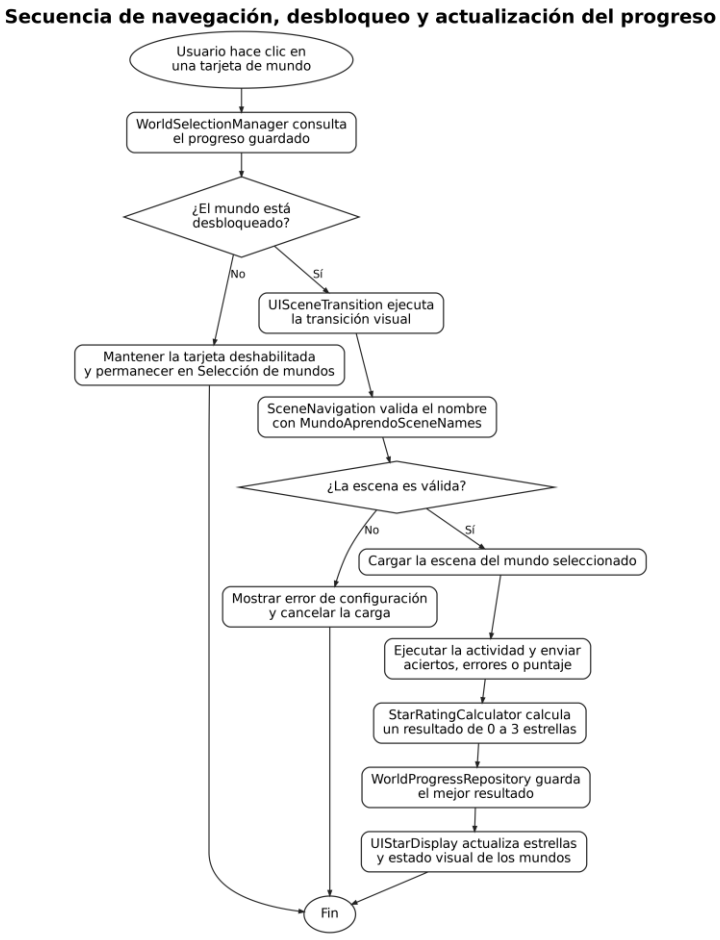
El punto de entrada de toda la cadena es el evento de clic de los botones de la interfaz, que invoca directamente un método público del controlador correspondiente. Por ello, la revisión de una escena debe empezar comprobando qué método está enlazado en cada botón y qué referencias tiene asignadas en el Inspector, ya que un enlace ausente no produce error de compilación y solo se manifiesta como un botón que no responde.

La Ilustración 39 sigue esa secuencia completa desde el evento de clic. En ella se distingue la intervención sucesiva del gestor de la selección, el repositorio de progreso, la

transición visual y el cargador de escenas, así como la resolución del nombre de la escena a partir de las constantes declaradas. El esquema incluye también el cierre de una actividad, con el cálculo de la calificación, el guardado del mejor resultado y la actualización de las estrellas en la interfaz, y representa las dos ramas de excepción previstas: el intento de abrir un mundo bloqueado y el intento de cargar una escena que no está registrada en la configuración de compilación.

Ilustración 39

Secuencia de navegación, desbloqueo y actualización del progreso.



Nota. Elaboración propia a partir de los scripts y de la configuración del proyecto.

La Ilustración 40 detalla la interacción del módulo de voz. El recorrido comienza en el controlador del cuento y en la comprobación de disponibilidad del micrófono y de los

permisos del sistema operativo. El diagrama distingue explícitamente los tres niveles implicados: el controlador, que solo conoce el contrato; la implementación para Windows, que lo cumple; y el componente del sistema operativo que esta encapsula. Se diferencian además los resultados parciales de los finales y las tres condiciones de cierre previstas, que son el silencio prolongado, el error del servicio y el tiempo máximo de escucha.

Descritos los tres mecanismos de integración, conviene recorrer el funcionamiento completo del videojuego como una sola cadena, porque es la forma en que un sistema puede comprobarse de extremo a extremo. El recorrido comienza al ejecutarse la aplicación, que carga la escena registrada en el índice cero de la configuración de compilación. En ella, el gestor de audio se inicializa como instancia única, recupera del almacenamiento local los volúmenes maestro y de música guardados en la sesión anterior y los aplica antes de que suene nada, de modo que el usuario no percibe un salto de volumen al arrancar.

Desde el menú principal, la opción de jugar invoca SceneNavigation mediante el evento On Click enlazado al botón. El cargador resuelve el destino a partir de MundoAprendoSceneNames, comprueba que la escena esté habilitada en la compilación y, si la comprobación es correcta, entrega el cambio visual a UISceneTransition antes de ejecutar la carga. De este modo, el efecto de fundido y la validación del destino permanecen en componentes diferentes: la transición puede sustituirse sin alterar las rutas y una ruta puede cambiarse sin reprogramar la animación.

Al abrirse la pantalla de selección, WorldSelectionManager consulta WorldProgressRepository antes de representar cada tarjeta. Para cada uno de los cuatro mundos, el repositorio responde si está desbloqueado y cuántas estrellas conserva como

mejor resultado. La disponibilidad sigue una progresión lineal: el primer mundo se habilita desde el inicio y los posteriores dependen del mundo anterior. Con esas respuestas, los componentes visuales deciden qué tarjetas pueden recibir interacción, cuáles muestran candado y qué estrellas deben aparecer, sin acceder directamente a las claves de PlayerPrefs.

Cuando el usuario pulsa una tarjeta, el gestor de la selección vuelve a comprobar el desbloqueo antes de hacer nada más. Si el mundo está bloqueado, no carga ninguna escena: muestra un mensaje temporal, reproduce la reacción visual de la tarjeta y devuelve el control. Si está desbloqueado, comprueba que tenga configurado un nombre de escena y delega en el mismo cargador utilizado desde el menú. Esa doble comprobación, en el dibujado y en la pulsación, evita que un estado desactualizado de la interfaz permita entrar donde no corresponde.

Cargada la escena de la actividad, su controlador prepara el estado inicial antes de que el usuario pueda intervenir: resuelve las referencias que falten, enlaza los botones, deja visible el panel inicial y bloquea la entrada. A continuación consulta la marca de tutorial de ese mundo. Si es la primera visita, activa la capa del tutorial y queda a la espera de su evento de finalización; si el tutorial ya se completó, la actividad queda lista de inmediato. Solo cuando esa decisión se resuelve se habilita la interacción.

Durante la actividad, cada entrada del usuario sigue el mismo trayecto. El evento de clic invoca un método del controlador; el controlador comprueba primero si está aceptando entrada, lo que descarta las pulsaciones producidas durante una animación o una demostración; aplica después la regla de validación propia de su mecánica; y traduce el resultado en tres efectos simultáneos: una reacción visual sobre el elemento pulsado, un

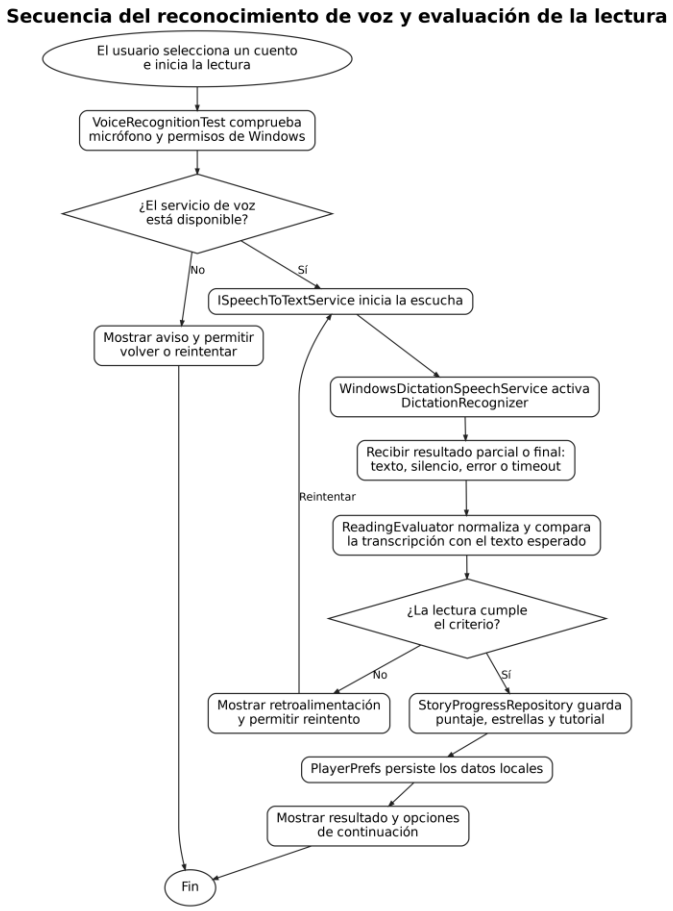
efecto sonoro solicitado al gestor de audio y la emisión de un evento de validación. Ese evento es el que recoge el puente del personaje guía para mostrar el mensaje de acierto o de reintento, sin que el controlador conozca su existencia.

Al completarse la última ronda o secuencia, el controlador cierra la actividad. Detiene las corrutinas en curso, bloquea la entrada, calcula la calificación según la regla de su mundo y entrega el resultado al repositorio correspondiente. WorldProgressRepository compara el valor con el mejor resultado histórico, conserva el mayor, marca el mundo como completado y emite ProgressChanged. En el Mundo de los Cuentos, StoryProgressRepository añade un nivel de detalle: conserva por separado el mejor puntaje y las estrellas de cada relato y comunica el avance general cuando se cumple la condición de cuentos completados. Solo después de actualizar la persistencia se presenta el panel final y la animación de estrellas, de modo que la interfaz siempre refleja un estado ya guardado.

El cierre del ciclo ocurre al regresar a la pantalla de selección. Sus componentes, que permanecían suscritos a la notificación del repositorio, ya han actualizado los bloqueos y las estrellas, de modo que el mundo recién completado aparece con su calificación y el siguiente ha dejado de mostrar el candado. Desde ahí el usuario puede repetir la actividad, entrar al mundo desbloqueado o volver al menú, y el recorrido vuelve a empezar. Comprobar esta cadena completa en una sola ejecución equivale a verificar simultáneamente la navegación, la persistencia, el cálculo de la calificación y la actualización de la interfaz, y por ello constituye el caso de prueba principal descrito en el apartado 4.2.11.

Ilustración 40

Secuencia del reconocimiento de voz y evaluación de la lectura.



Nota. Elaboración propia a partir de los scripts y de la configuración del proyecto.

4.2.8 Integración entre diseño, arte y programación

La integración técnica comienza al asignar sprites, textos, botones y animadores a los objetos de la escena. Los eventos On Click llaman al controlador, que actualiza paneles, sonidos y estrellas; después, los repositorios guardan el mejor resultado y notifican a SeleccionMundos. Esta relación entre recurso, componente de Unity y script debe comprobarse cada vez que se sustituya un asset o se cambie una referencia.

El pipeline artístico descrito en el primer bloque condicionó esta integración. Las poses se exportaron como PNG independientes precisamente porque los controladores

visuales de Unity trabajan con referencias concretas a Sprite. Cuando una respuesta cambia el estado del personaje, el script no modifica el dibujo: sustituye la referencia por otra pose ya preparada. Esta decisión mantiene el arte fuera de la lógica y permite que una pose sea corregida en Photoshop y reemplazada sin reescribir la mecánica.

Los assets de interfaz adaptados desde Cartoon UI, Hyper_Casual_UI y Space_Exploration_GUI_Kit se integraron de la misma manera que los recursos propios una vez modificados. El origen del asset no cambia su comportamiento dentro de Unity: después de la adaptación visual se importa como recurso del proyecto y se vincula a Image, Button u otros componentes. La diferencia documental se conserva en la procedencia y en la licencia, no en el flujo de ejecución.

La elección de referencias serializadas desde el Inspector permite que la relación entre arte y programación sea visible. Un controlador puede exponer un Sprite de estado, un Image de personaje, un AudioClip o un RectTransform sin localizarlo por nombre durante cada ejecución. Esto reduce búsquedas dinámicas, facilita detectar referencias faltantes al revisar la escena y permite que un cambio artístico se concentre en el campo correspondiente en lugar de obligar a modificar el código.

La interfaz de Mundo Aprendo se construyó íntegramente con el sistema de interfaz de usuario de Unity, sin elementos dibujados en el mundo tridimensional. Cada una de las seis escenas contiene un único lienzo configurado en modo de superposición sobre la pantalla, lo que significa que la interfaz se dibuja siempre por encima de la escena y no depende de la posición de una cámara. Esa decisión simplifica el proyecto: al no existir perspectiva, la posición de un botón en el editor coincide exactamente con la que percibe el usuario.

El escalado se resolvió con el componente de ajuste del lienzo en modo de escalado según el tamaño de pantalla, tomando como resolución de referencia 1920 por 1080 píxeles y con el criterio de correspondencia repartido por igual entre anchura y altura. La combinación es deliberada: la aplicación se compila con una resolución base de 1024 por 768 y ventana no redimensionable, de modo que la interfaz se diseñó a una resolución mayor y se reduce proporcionalmente al ejecutarse. Repartir el criterio de correspondencia por igual evita que un cambio de proporción recorte los extremos horizontales o verticales, que es el riesgo habitual cuando se prioriza una sola dimensión.

Los anclajes siguen dos patrones según la función del elemento. La mayoría de los objetos se anclan al centro del lienzo, lo que los mantiene agrupados alrededor del eje de la pantalla cuando la resolución cambia. Los elementos que deben permanecer pegados a un borde, como los fondos y los contenedores a pantalla completa, se anclan a la esquina inferior izquierda y se extienden. Existe además un tercer grupo, propio de la pantalla de selección: las tarjetas de los mundos emplean anclajes en posiciones fraccionarias distintas entre sí, lo que las distribuye de forma proporcional sobre el fondo y conserva su disposición relativa al escalar.

Los textos se resolvieron con TextMesh Pro en lugar del componente de texto heredado, decisión que responde a que el escalado de la interfaz obliga a redimensionar la tipografía y el renderizado por campo de distancia mantiene los bordes definidos donde el texto tradicional aparecería borroso. La densidad de texto es desigual entre escenas y refleja su complejidad: el Mundo de los Cuentos concentra la mayor cantidad de componentes de texto, seguido por el Mundo Musical, mientras que el Mundo de los Tamaños es el más ligero por apoyarse principalmente en imágenes.

La organización interna de cada escena se apoya en paneles que se activan y desactivan en lugar de cargarse por separado. Las actividades siguen todas el mismo esquema de tres paneles: uno inicial con la consigna y el botón de comienzo, uno de juego con los elementos interactivos y uno de resultado con las estrellas obtenidas. El controlador de cada mundo conserva referencias a los tres y alterna su visibilidad según la fase. Mantenerlos en la misma escena evita recargas intermedias y permite que la transición entre fases sea inmediata, a costa de que todos los elementos existan en memoria desde el arranque, coste asumible dado el tamaño de las escenas.

Los estados visuales de los botones se resolvieron de dos maneras. La mayor parte emplea la transición por tinte de color, que Unity aplica automáticamente a los estados normal, resaltado, presionado, seleccionado y deshabilitado. Un grupo reducido del menú principal utiliza transición por animación, al tratarse de los planetas que actúan como botones y requieren un movimiento propio. Un tercer grupo tiene la transición desactivada, y corresponde a los elementos cuyo comportamiento visual gobierna un script propio: las tarjetas de mundo de la pantalla de selección y las teclas del piano, que necesitan combinar el estado de bloqueo con la retroalimentación de acierto o error y no pueden delegarlo en el tinte automático.

La conexión entre la interfaz y la lógica se establece mediante los eventos de clic configurados en el Inspector, que invocan directamente métodos públicos de los controladores. Esa asignación tiene una consecuencia práctica para el mantenimiento: los métodos enlazados desde el Inspector no aparecen como referenciados al buscarlos en el código, de modo que renombrarlos rompe el enlace sin producir ningún error de compilación. El fallo solo se manifiesta como un botón que deja de responder. Por ese

motivo, los controladores conservan nombres de método estables y varios de ellos vuelven a enlazar sus botones por código durante la inicialización, únicamente cuando detectan que el evento no tiene ninguna llamada persistente configurada.

Cada escena incorpora además un sistema de eventos con el módulo de entrada del paquete Input System, que es el que traslada el clic del ratón hasta el botón correspondiente. Sin ese objeto la interfaz se dibuja pero no responde, por lo que su presencia forma parte de las comprobaciones al crear una escena nueva.

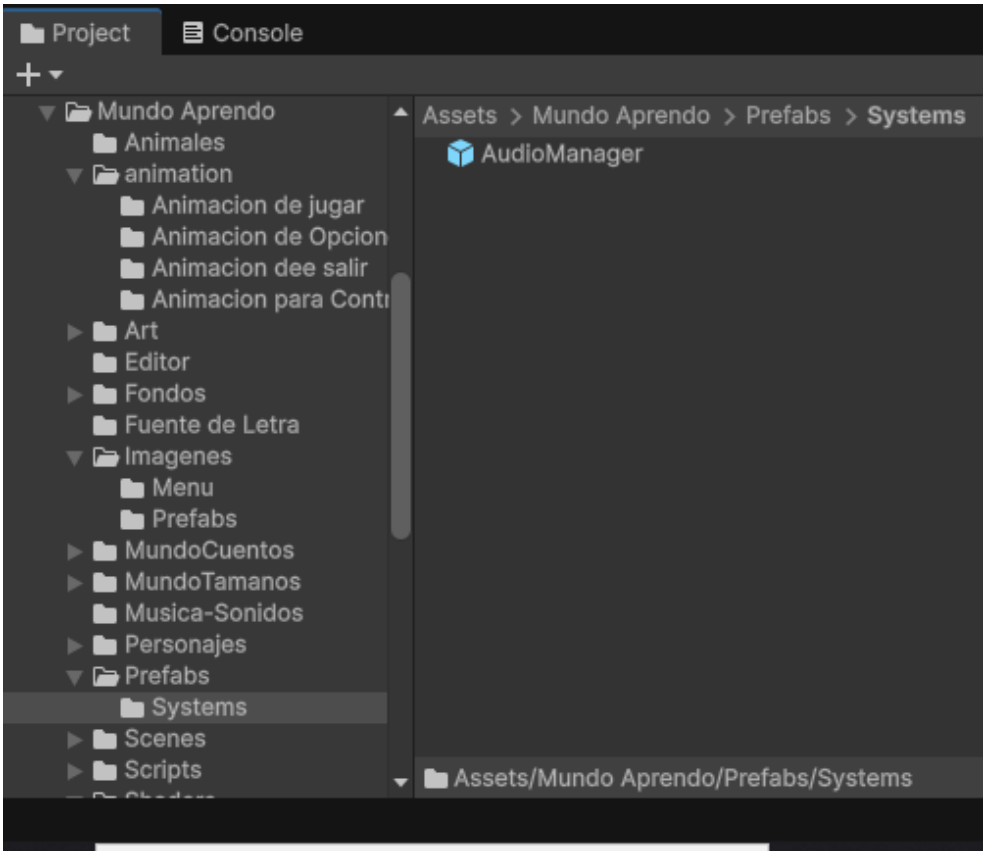
Ilustración 41

Organización de la carpeta UI del proyecto.

Nota. Captura propia del proyecto y de sus registros técnicos.

Ilustración 42

Prefab del sistema de audio.



Nota. Captura propia del proyecto y de sus registros técnicos.

4.2.9 Gestión de datos y persistencia

La persistencia del proyecto se resolvió íntegramente en local, sin base de datos ni servicio remoto. El mecanismo principal es el sistema de preferencias de Unity, que almacena pares de clave y valor y que, en la compilación para Windows, los conserva en el registro del sistema operativo. La Tabla 20 registra las claves verificadas y su función.

Las claves se construyen dentro del repositorio a partir de un formato único que incorpora el índice del mundo, comprendido entre cero y tres. Cada mundo emplea dos claves: una que indica si fue completado y otra que conserva su mejor número de estrellas. A ellas se añaden las marcas de tutorial, una por cada mundo que dispone de uno, que

permiten mostrar las instrucciones únicamente en la primera visita. El Mundo de los Cuentos amplía este esquema con claves propias por cada cuento, que registran si se completó, su mejor puntaje y sus mejores estrellas, junto con un listado de los identificadores de cuento ya registrados; ese listado resulta necesario porque sin él no sería posible localizar y borrar todas las claves individuales al reiniciar el progreso.

Cada escritura se confirma de forma explícita en lugar de dejarla al cierre de la aplicación. Unity solo garantiza el volcado de las preferencias cuando el programa termina de manera ordenada, de modo que forzar el guardado tras cada cambio evita perder el avance si el proceso se interrumpe de forma anómala. El coste de esa decisión es una escritura adicional al terminar cada actividad, irrelevante frente al riesgo de perder el progreso de una sesión.

Existe una excepción al mecanismo general. Las preferencias de pantalla no se guardan en el sistema de preferencias, sino en un archivo JSON alojado en el directorio de datos persistentes de la aplicación. La diferencia responde a que la configuración de vídeo agrupa varios valores relacionados entre sí —resolución, modo de pantalla, sincronización vertical y límite de fotogramas— que resultan más manejables como un objeto único que como claves sueltas.

El reinicio del progreso está previsto de forma explícita. La pantalla de selección ofrece esa opción y, al confirmarse, el repositorio elimina las claves de los cuatro mundos, las de todos los cuentos registrados y las marcas de tutorial de cada mundo, y confirma el borrado en una sola operación de escritura. Tras ello emite su notificación de cambio, de manera que la interfaz vuelve al estado inicial sin recargar la escena.

Conviene señalar las limitaciones de este enfoque para no interpretarlas como omisiones. Los datos son locales al equipo y a la cuenta de usuario del sistema operativo, por lo que no acompañan al jugador entre máquinas ni distinguen perfiles distintos dentro de una misma sesión de Windows. Tampoco existe cifrado ni verificación de integridad, de modo que un usuario con acceso al registro puede alterar el progreso. Para el alcance del prototipo, orientado a un uso local y con acompañamiento adulto, estas condiciones resultan admisibles; una versión destinada a varios usuarios simultáneos exigiría introducir perfiles y un almacenamiento distinto.

Tabla 20

Datos y persistencia verificados.

Dato	Mecanismo	Lectura o escritura	Estado verificado
Mundo completado	MundoAprendo_World_{índice}_Completed	Lectura y escritura con PlayerPrefs	Implementado
Mejores estrellas	MundoAprendo_World_{índice}_Stars	Conserva el mejor valor entre 0 y 3	Implementado
Progreso de cuentos	StoryProgressRepository	Completado, puntaje, estrellas y tutorial	Implementado
Micrófono	selected_microphone_device	Guarda el nombre del dispositivo	Implementado con limitación en Windows
Preferencias de pantalla	screenSettings.json	Archivo local en el directorio persistente de la aplicación	Implementado
Base de datos o servicio remoto	No utilizado	No aplica	Fuera del alcance

Nota. Elaboración propia a partir del proyecto y sus registros técnicos.

Para asegurar reproducibilidad, la entrega final debe registrar cada clave de PlayerPrefs, su valor predeterminado, el momento de lectura y escritura, y el procedimiento de reinicio. También debe documentar la ubicación de screenSettings.json, su recuperación ante corrupción y el comportamiento después de cerrar, reabrir o restablecer el progreso.

4.2.10 Configuración de escenas y flujo del videojuego

La compilación de Windows debe conservar Mundo aprendo tesis.exe junto con Mundo aprendo tesis_Data, MonoBleedingEdge, UnityPlayer.dll y las bibliotecas generadas por Unity. Separar el ejecutable de estos elementos impide reproducir la versión documentada.

Para comprobar el inicio, ejecute el archivo desde la carpeta Build y confirme la carga de Menu. Desde JUGAR, la navegación debe abrir SeleccionMundos y mantener disponibles las opciones de retorno.

Antes de continuar con las mecánicas, compruebe que la salida de audio funcione y que no existan referencias ausentes en la consola o en el Inspector.

Para probar MundoCuentos, habilite el micrófono predeterminado y los permisos de voz de Windows; una configuración diferente debe registrarse en el caso de prueba.

El proyecto se organiza en seis escenas registradas en la configuración de compilación con índices consecutivos del cero al cinco. Sus nombres no se escriben de forma literal en ningún controlador: se declaran una sola vez como constantes en un archivo dedicado, y el resto del código las referencia a través de ellas. Esa restricción tiene una consecuencia práctica comprobada durante el desarrollo, cuando la escena de cuentos se renombró: bastó modificar la constante para que todas las transiciones que apuntaban a ella siguieran funcionando. La Tabla 21 conserva el inventario verificado.

El flujo entre escenas es cerrado y siempre reversible. El menú conduce a la pantalla de selección; la selección abre uno de los cuatro mundos, siempre que esté desbloqueado; y cada mundo ofrece el retorno a la selección desde su interfaz y desde su pantalla de resultado. El Mundo Musical incorpora además una transición directa al mundo siguiente

desde el panel de cierre, de modo que un usuario que avanza sin interrupciones no necesita volver al selector entre las dos primeras actividades.

Cada escena sigue el mismo patrón de inicialización, apoyado en el ciclo de vida de los componentes de Unity. En la primera fase, el controlador resuelve las referencias que falten, enlaza los botones a sus métodos y deja la actividad en un estado de reposo, con el panel inicial visible y la entrada bloqueada. En la segunda fase decide si corresponde mostrar el tutorial. Al desactivarse o destruirse, el controlador detiene las corrutinas en curso, interrumpe el audio y cancela las suscripciones a eventos, lo que evita que una secuencia iniciada en la escena anterior siga modificando una interfaz que ya no existe.

Los tutoriales condicionan el arranque de tres de las cuatro actividades. El controlador consulta la marca guardada para ese mundo: si el tutorial ya se completó, la actividad queda preparada de inmediato; si no, la capa del tutorial se activa y el controlador espera su evento de finalización antes de habilitar la entrada. Las capas de tutorial se dejan desactivadas en la escena de forma deliberada, para que no aparezcan durante una fracción de segundo mientras la escena carga.

Incorporar una escena nueva al proyecto exige tres pasos que deben realizarse en conjunto: declarar su nombre como constante, registrarla en la configuración de compilación y comprobar las transiciones que la enlazan en ambos sentidos. Omitir el segundo paso no impide compilar, pero provoca que la carga falle en tiempo de ejecución; por eso el cargador de escenas verifica previamente que la escena esté habilitada y deja constancia en la consola en lugar de interrumpir la ejecución.

Tabla 21

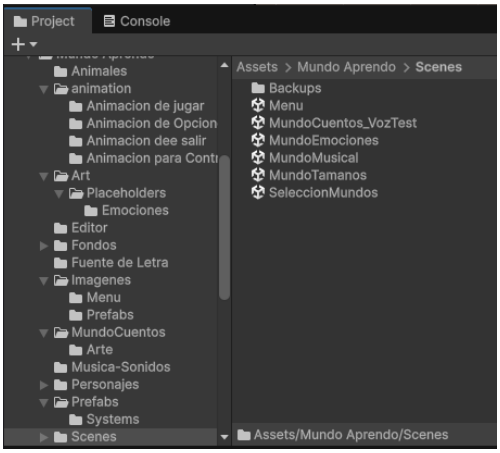
Escenas y flujo configurados.

Índice de compilación	Escena	Función	Transición principal
0	Menu	Presentar Jugar, Controles, Opciones y Salir.	Hacia SeleccionMundos.
1	SeleccionMundos	Mostrar cuatro mundos, bloqueos y estrellas.	Hacia el mundo elegido o Menu.
2	MundoMusical	Ejecutar secuencias musicales.	Retorno a SeleccionMundos.
3	MundoCuentos	Ejecutar lectura apoyada por voz.	Retorno a SeleccionMundos.
4	MundoTamanos	Ejecutar comparaciones de animales.	Retorno a SeleccionMundos.
5	MundoEmociones	Ejecutar selección de emociones.	Retorno a SeleccionMundos.

Nota. Elaboración propia a partir del proyecto y sus registros técnicos.

Ilustración 43

Escenas del proyecto.



Nota. Captura propia del proyecto y de sus registros técnicos.

Ilustración 44

Jerarquía de la escena SeleccionMundos.

Nota. Captura propia del proyecto y de sus registros técnicos.

4.2.11 Pruebas técnicas y validación

La comprobación técnica del prototipo se organizó mediante pruebas automatizadas disponibles en el proyecto y recorridos funcionales sobre la aplicación. Su propósito es verificar comportamiento de software: carga de escenas, respuesta de controles, ejecución de reglas, persistencia, voz y compilación. Estas comprobaciones no evalúan aprendizaje, aceptación infantil ni efectos cognitivos y, por tanto, sus resultados se interpretan exclusivamente como evidencia de implementación.

La unidad de evaluación fue el propio prototipo: escenas, componentes, reglas, datos persistentes, servicio de voz y compilación. Los criterios de resultado fueron técnicos y se expresaron como aprobación o fallo de casos, presencia de errores o advertencias y generación de la compilación. No se aplicaron instrumentos a personas ni se interpretaron estos registros como mediciones de usabilidad, comprensión, aceptación o aprendizaje.

Las pruebas automatizadas conservadas en la versión revisada se ejecutan en modo de reproducción y están separadas del código distribuible mediante su propia definición de ensamblado. Esta organización permite comprobar escenas y componentes dentro del entorno de Unity sin incorporar los archivos de prueba a la compilación final. Los casos se utilizan como apoyo para detectar referencias ausentes, configuraciones inconsistentes y cambios que puedan afectar el recorrido del prototipo.

La batería disponible cubre la estructura de escenas, la navegación, la selección de mundos, el comportamiento de las actividades y partes del módulo de voz. Las verificaciones de escena comprueban la presencia de componentes indispensables; las de navegación revisan que los destinos puedan cargarse; y las de los minijuegos comprueban estados de tutorial, entradas, validación y cierre. En el Mundo Musical se revisan además las siete teclas, la reproducción de secuencias y el avance entre estados de la actividad.

El Mundo Musical concentra varios casos porque combina temporización, reproducción, bloqueo de entrada y comparación de una secuencia ordenada. Separar estas comprobaciones permite identificar si un problema pertenece a la configuración de las teclas, a la reproducción de la secuencia o a la validación de la respuesta, en lugar de tratar la actividad como una única operación indivisible.

En el módulo de voz, parte de la lógica puede comprobarse sin depender del micrófono. Los filtros de vocabulario, normalización y eliminación de fragmentos repetidos pueden recibir texto controlado y verificar su salida de manera determinista. Esta estrategia es útil porque separa los errores de procesamiento textual de las condiciones variables del hardware y del servicio de dictado.

Los registros técnicos se seleccionaron por su relación directa con el comportamiento documentado. Se conservaron las evidencias que permiten identificar la ejecución, la función comprobada y el resultado observado, evitando mezclar capturas que respondan a configuraciones distintas. De esta forma, la documentación vincula cada afirmación del manual con una evidencia concreta de navegación, mecánicas, persistencia, voz o compilación.

La Tabla 22 resume las evidencias técnicas utilizadas para documentar el estado funcional: ejecuciones automatizadas sin fallos en las revisiones registradas, una prueba de voz sin errores visibles en consola y una compilación de Windows completada correctamente. En conjunto, estos registros respaldan la integración de escenas, controles, reglas, persistencia y servicios dentro del prototipo.

La segunda vía de comprobación es el recorrido manual sobre la compilación, descrito paso a paso en la Tabla 23. Su finalidad es atravesar en una sola ejecución la cadena que conecta menú, selector, minijuego, validación, guardado y retorno. Este recorrido complementa las pruebas aisladas porque comprueba que componentes que funcionan por separado también intercambien correctamente el control y los datos cuando el usuario sigue el flujo completo.

La verificación funcional se interpreta dentro del alcance técnico definido para la tesis. Por ello, los registros se utilizan para comprobar navegación, respuesta de controles, ejecución de reglas, persistencia, voz y compilación, sin convertirlos en una evaluación pedagógica ni en una medición de aprendizaje. Esta separación permite presentar con precisión lo que el software hace y mantener fuera del análisis los efectos que requerirían un estudio posterior con usuarios.

Tabla 22

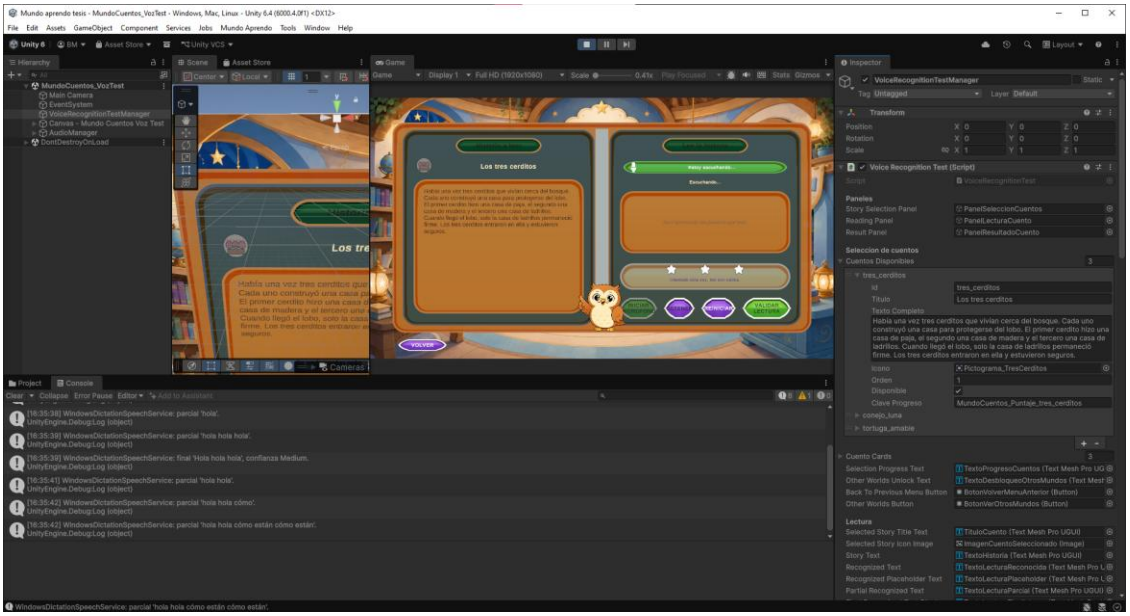
Resultados técnicos registrados durante la comprobación.

Evidencia	Fecha	Resultado observado	Estado
EditMode registrado	Jun-26	33 aprobadas, 0 fallidas, 0 omitidas.	Evidencia válida para la revisión registrada.
PlayMode registrado	Jun-26	8 aprobadas, 0 fallidas, 0 omitidas.	Evidencia válida para la revisión registrada.
Consola de prueba de voz	Captura disponible	0 errores y 1 advertencia visible.	Evidencia del funcionamiento técnico mostrado.
Compilación Windows	Jul-26	Build completed with a result of Succeeded.	Compilación correcta.

Nota. Elaboración propia a partir del proyecto y sus registros técnicos.

Ilustración 45

Consola durante la prueba de voz: cero errores y una advertencia visible.



Nota. Captura propia del proyecto y de sus registros técnicos.

Una ejecución reproducible debe registrar idioma de Windows, modelo de micrófono, permisos, distancia aproximada, ambiente, frases previstas, variantes aceptadas, tiempo de espera y respuesta observada. Los casos deben cubrir reconocimiento correcto, transcripción parcial, silencio, rechazo y reintento. El resultado debe asociarse con fecha, versión del proyecto y equipo de prueba.

El módulo de voz debe verificarse en el entorno Windows de entrega porque depende del micrófono, de permisos y del servicio de dictado. Para que la comprobación sea reproducible conviene registrar idioma del sistema, dispositivo utilizado y condiciones básicas de escucha. Si posteriormente se amplía la compatibilidad a otros equipos o servicios, esa validación deberá repetirse sin modificar el criterio de evaluación definido por ReadingEvaluator.

Las pruebas de cierre deben comprobar también que cada actividad produzca un resultado válido, que el repositorio conserve el mejor valor y que el selector refleje ese cambio al regresar. Esta comprobación enlaza el comportamiento local del minijuego con los servicios compartidos de progreso y constituye una evidencia más completa que revisar únicamente el panel de resultado.

La comprobación de extremo a extremo se ejecuta como un caso técnico desde la carga de Menu hasta la actualización del progreso. En cada paso se identifica la escena, el componente responsable y el resultado esperado. La Tabla 23 define la secuencia mínima y permite repetirla después de cada cambio que afecte navegación, guardado, estrellas o carga de escenas.

Tabla 23

Secuencia técnica de comprobación.

Paso	Entrada o acción de prueba	Comprobación técnica	Evidencia
1	Iniciar la compilación desde la carpeta Build.	Menu carga sin archivos faltantes.	Ilustración 1.
2	Activar JUGAR.	SceneNavigation abre SeleccionMundos y sus referencias responden.	Ilustraciones 5, 7 y 8.
3	Abrir un mundo habilitado.	Se cargan el tutorial, el controlador y los recursos de la escena.	Ilustraciones 6 y 9-12.
4	Ejecutar una entrada válida y una no válida.	El controlador aplica validación, retroalimentación y reintento.	Ilustraciones 9-13.
5	Completar la actividad y volver al selector.	Se calcula el resultado, se guarda el mejor valor y se actualizan las estrellas.	Ilustraciones 13 y 22.

Nota. Elaboración propia a partir de capturas del proyecto.

Las evidencias registradas permiten documentar que los sistemas principales han sido integrados y que existe un procedimiento reproducible para recorrerlos. La secuencia de la Tabla 23 enlaza menú, selección de mundos, actividad, validación, guardado y

retorno, por lo que sirve como comprobación de extremo a extremo y como guía de mantenimiento cuando se modifique una escena o un sistema compartido.

4.2.12 Optimización y rendimiento

La optimización del prototipo se abordó como un conjunto de decisiones de configuración y organización del código adoptadas durante el desarrollo. El objetivo fue evitar trabajo innecesario durante la ejecución, mantener controladas las referencias entre objetos y conservar una presentación estable en las resoluciones documentadas. Por ello, este apartado explica las decisiones aplicadas al renderizado, la interfaz, el acceso a datos, las corrutinas y la generación de audio.

En el plano gráfico, el proyecto emplea el pipeline universal de renderizado con dos perfiles de calidad definidos. Al tratarse de un videojuego bidimensional compuesto por elementos de interfaz y sprites, la carga de renderizado es reducida y no requirió intervención específica. Los sprites del proyecto se importan con transparencia habilitada, filtrado bilineal y una resolución máxima de 2048 píxeles, valores predeterminados que resultaron suficientes para el formato de trabajo empleado.

En el plano del código, la decisión con mayor efecto sobre el rendimiento fue evitar las búsquedas dinámicas de objetos durante la ejecución. Las referencias se asignan desde el Inspector mediante campos serializados y se resuelven una sola vez durante la inicialización, en lugar de localizarse en cada fotograma. Del mismo modo, la lectura del progreso se realiza al abrirse la pantalla de selección y al recibir la notificación de cambio, y no de forma continua, lo que evita accesos repetidos al almacenamiento local. Las animaciones se resuelven mediante corrutinas que se detienen explícitamente al

desactivarse el componente, de manera que una secuencia iniciada en una escena no continúe ejecutándose tras cambiar de escena.

Una decisión menos evidente afecta al audio del Mundo Musical. Los tonos generados por software se conservan en un diccionario indexado por frecuencia, de modo que cada tono se sintetiza una única vez y las pulsaciones posteriores reutilizan el clip almacenado. Sin ese almacenamiento intermedio, cada pulsación implicaría generar más de diez mil muestras de audio en tiempo real.

Las comprobaciones de interfaz dejaron evidencia verificable. Dos casos de la batería automatizada confirman que los controles activos permanecen dentro del lienzo en las resoluciones exigidas, lo que constituye una comprobación del escalado descrito en el apartado 4.2.8. El proyecto no atribuye a estas pruebas una medición cuantitativa de rendimiento, sino una verificación funcional de la presentación de la interfaz.

Estas decisiones de optimización se relacionan con el tipo de aplicación desarrollada: un videojuego 2D basado principalmente en sprites, Canvas y controladores de eventos. La configuración evita búsquedas repetidas, reutiliza datos ya calculados y detiene procesos visuales cuando dejan de ser necesarios. Si en futuras versiones aumenta la cantidad de escenas, animaciones o recursos simultáneos, Unity Profiler puede utilizarse como apoyo para comparar consumo de CPU, memoria y tiempos de carga.

4.2.13 Despliegue y compilación

El despliegue corresponde a la plataforma Windows Standalone de 64 bits con backend Mono. La configuración del reproductor fija una resolución base de 1024 por 768 píxeles, arranque en pantalla completa y ventana no redimensionable; esta última condición es coherente con el escalado de interfaz descrito en el apartado 4.2.8, ya que descarta las

proporciones arbitrarias que produciría una ventana ajustable por el usuario. La eliminación de código de motor no utilizado se encuentra habilitada.

La compilación registra las seis escenas en el orden descrito en la Tabla 21, y ese orden importa más allá de la organización: la escena situada en el índice cero es la que se carga al ejecutar la aplicación. Alterar ese orden cambiaría el punto de entrada del videojuego sin producir ningún error de compilación.

La carpeta de compilación resultante ocupa alrededor de 302 megabytes. El reparto es revelador: los datos del proyecto concentran unos 251 megabytes y el motor de Unity aporta unos 35 megabytes en su biblioteca principal, mientras que el ejecutable propiamente dicho apenas supera el medio megabyte. Esa proporción explica por qué el ejecutable no puede distribuirse aislado: contiene únicamente el punto de entrada, y sin la carpeta de datos, la biblioteca del motor y el entorno de ejecución administrado no dispone de escenas, recursos ni código que ejecutar.

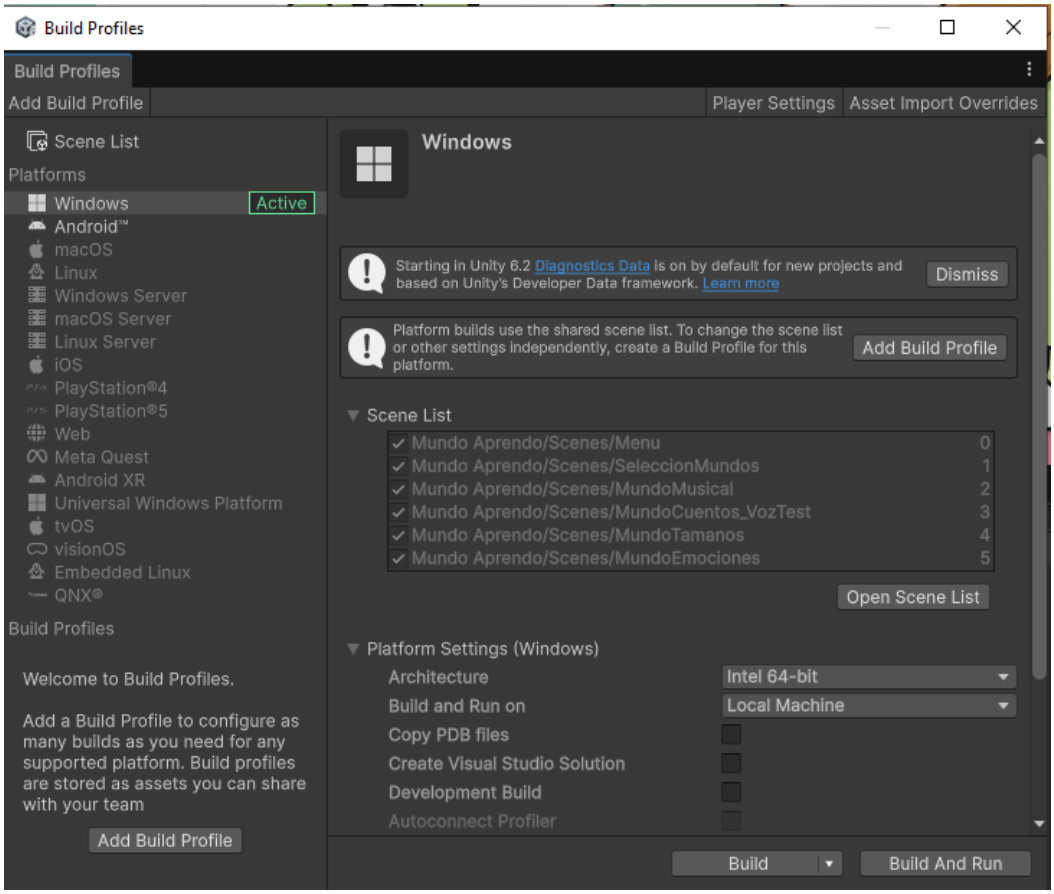
La distribución para Windows conserva el ejecutable junto con la carpeta de datos, las bibliotecas de Unity y los componentes requeridos por el entorno de ejecución. Esta estructura debe mantenerse completa al trasladar la aplicación a otro equipo, ya que el ejecutable actúa como punto de entrada y depende de los archivos generados durante la compilación para cargar escenas, recursos y código.

El procedimiento de comprobación del despliegue consiste en ejecutar la aplicación desde la carpeta de compilación, no desde el editor, y confirmar que el menú carga sin referencias ausentes en la consola. Esa distinción es relevante porque el editor resuelve algunos recursos que la compilación exige tener correctamente registrados, de modo que un proyecto que funciona en el editor puede fallar compilado. Las Ilustraciones 46 y 47

documentan la configuración del perfil de compilación y el contenido resultante de la carpeta.

Ilustración 46

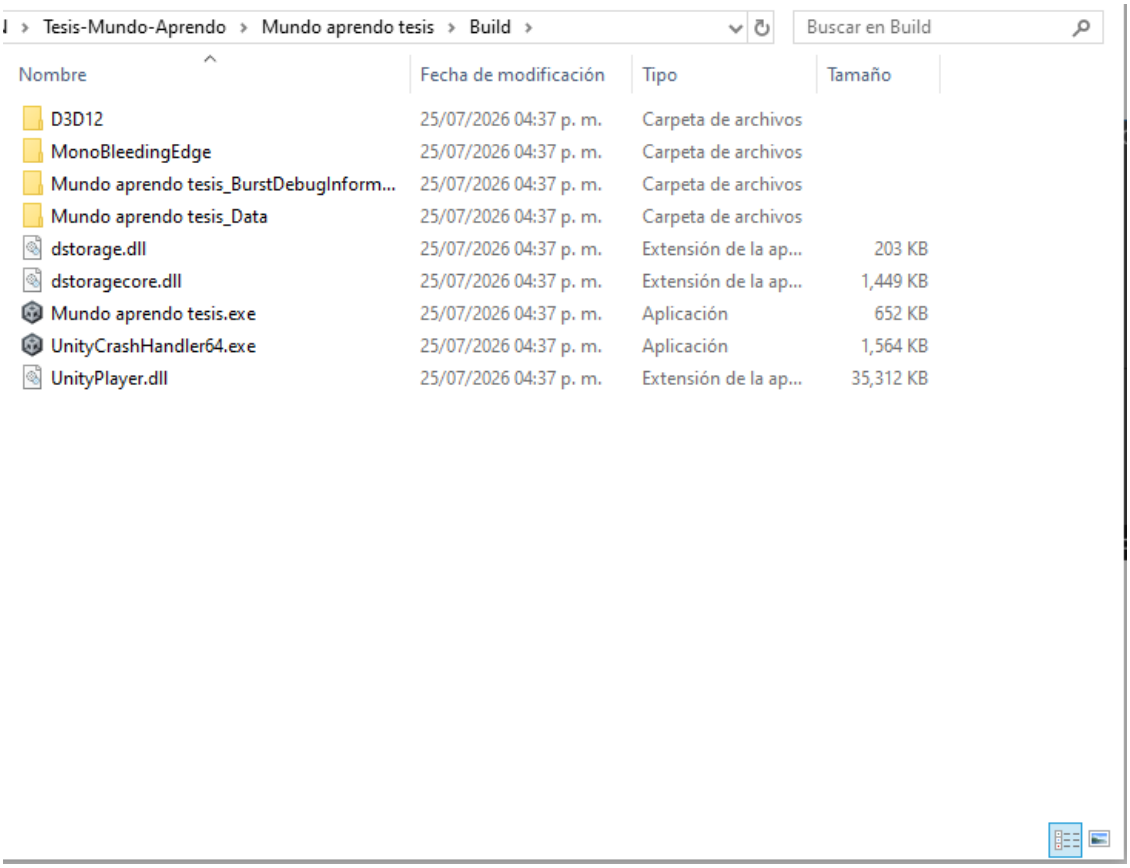
Configuración del perfil de compilación para Windows.



Nota. Captura propia del proyecto y de sus registros técnicos.

Ilustración 47

Contenido de la carpeta Build y archivo ejecutable de Mundo Aprendo.



Nota. Captura propia del proyecto y de sus registros técnicos.

4.2.14 Escalabilidad y mantenimiento

La capacidad de ampliación del proyecto depende de tres decisiones ya descritas: la centralización de los nombres de escena, la concentración de la persistencia en un repositorio único y la comunicación por eventos entre los mundos y la interfaz. Este apartado traduce esas decisiones en consecuencias concretas de mantenimiento.

Incorporar un quinto mundo exige intervenir cuatro puntos y solo cuatro. Debe declararse el nombre de la escena como constante nueva; debe elevarse el número total de mundos que reconoce el repositorio de progreso, ya que ese valor gobierna la validación de índices y el borrado al reiniciar; debe registrarse la escena en la configuración de

compilación; y debe añadirse su tarjeta a la pantalla de selección con el índice correspondiente. El controlador del mundo nuevo no necesita conocer a los anteriores: le basta con invocar el repositorio con su propio índice al terminar la actividad.

La regla de desbloqueo, en cambio, sí impone un límite. Está implementada como una progresión lineal en la que cada mundo depende del inmediatamente anterior, y esa relación está escrita dentro del repositorio. Un esquema distinto, como permitir varios mundos iniciales o exigir un número mínimo de estrellas acumuladas, requeriría modificar esa regla. La ventaja es que se encuentra en un solo lugar y no repartida entre los cuatro mundos.

El mantenimiento de una mecánica existente está acotado por el mismo criterio. Los parámetros que gobiernan la dificultad —número de rondas, pausas entre notas, umbrales de similitud de la lectura, umbrales de errores— están expuestos como campos serializados y pueden ajustarse desde el Inspector sin recompilar ni tocar el código. Esa exposición fue deliberada, porque son los valores que con mayor probabilidad requerirán ajuste tras una prueba con usuarios reales.

Para facilitar cambios futuros, el mantenimiento debe conservar la centralización de los nombres de escena, las reglas de progreso y las constantes utilizadas por los sistemas compartidos. Mantener estas decisiones en componentes específicos evita que una modificación de navegación, estrellas o persistencia obligue a editar múltiples escenas y permite que las pruebas sigan verificando el mismo comportamiento sin duplicar lógica.

El mantenimiento debe asociar cada entrega con una versión identificable del proyecto, conservar el origen y la configuración de los recursos visuales, documentar cambios en escenas, scripts y claves de persistencia y actualizar las pruebas afectadas.

Cada entrega debe actualizar el número de versión, el inventario de recursos, las dependencias, las escenas, las claves de persistencia, los casos de prueba y sus evidencias. Los criterios para incorporar assets permanecen en 4.1.12 y el procedimiento para ampliar mundos o sistemas en 4.10.6; este apartado registra los cambios y comprueba que la documentación siga correspondiendo con la compilación entregada.

4.2.15 Procedimiento para ampliar el sistema

El procedimiento para añadir una actividad nueva se describe a continuación en el orden en que debe ejecutarse, ya que cada paso depende del anterior. Se presenta como secuencia porque su valor reside precisamente en el orden: invertir dos pasos produce errores que solo se manifiestan en tiempo de ejecución.

El punto de partida son los recursos gráficos, preparados según los criterios del apartado 4.1.12, porque la escena no puede montarse sin ellos. Con los recursos disponibles se crea la escena y se replica su estructura común: un lienzo en superposición con la misma resolución de referencia, el sistema de eventos, la capa de transición visual y los tres paneles de inicio, juego y resultado. La batería de pruebas comprueba esos elementos en las seis escenas existentes, de modo que omitir cualquiera de ellos hará fallar el caso correspondiente.

A continuación, se escribe el controlador de la actividad, que debe asumir el mismo reparto de responsabilidades que los cuatro existentes: preparar el estado inicial, habilitar la entrada, validar la respuesta, calcular la calificación y entregarla al repositorio con su índice de mundo. Conviene que exponga también los eventos de validación y de finalización, ya que sin ellos el personaje guía no podrá reaccionar y la actividad quedará desconectada del sistema de retroalimentación.

Finalmente se integra la actividad en el flujo general: se declara la constante con el nombre de la escena, se registra la escena en la configuración de compilación, se amplía el número de mundos del repositorio y se añade la tarjeta a la pantalla de selección enlazando su botón al método de selección con el índice correspondiente. La comprobación mínima consiste en recorrer la cadena completa descrita en el apartado 4.2.7: entrar desde el selector, completar la actividad, verificar que las estrellas se guardan y confirmar que al regresar la tarjeta refleja el resultado y el mundo siguiente aparece desbloqueado.

4.2.16 Evidencias técnicas relevantes

Las evidencias incorporadas al capítulo cumplen funciones distintas y no son intercambiables entre sí. Las capturas del editor documentan configuraciones reales que no podrían deducirse del código, como la versión del motor, la organización de carpetas, la jerarquía de una escena o el enlace de un evento en el Inspector. Los diagramas representan flujos de ejecución que ninguna captura puede mostrar, porque describen una secuencia en el tiempo y no un estado. Las tablas sintetizan inventarios verificables. Y el texto explica el funcionamiento, que es lo único que ninguno de los tres formatos anteriores puede sustituir. La Tabla 23 identifica la evidencia que respalda cada aspecto.

El criterio aplicado fue que ninguna captura se reutilizara para justificar dos afirmaciones distintas. Una imagen de la consola durante una prueba de voz demuestra que esa prueba se ejecutó sin errores, pero no demuestra que el reconocimiento funcione con cualquier hablante ni en cualquier entorno; una captura de la carpeta de compilación demuestra que la compilación se generó, pero no que el ejecutable se comporte correctamente. Mantener esa correspondencia estricta entre evidencia y afirmación es lo que permite que el capítulo sea auditable.

Las evidencias disponibles cumplen funciones diferentes dentro del manual. Las capturas documentan configuraciones concretas del editor y resultados observables; los diagramas representan relaciones y secuencias que no pueden mostrarse en una sola captura; y las tablas sintetizan inventarios o estados verificables. Esta separación evita utilizar una misma imagen para justificar afirmaciones distintas y permite que la explicación escrita se concentre en el funcionamiento del sistema y en la relación entre sus componentes.

Tabla 24

Índice de evidencias técnicas incorporadas.

Evidencia	Finalidad documental	Archivos incorporados o estado
Interfaz y recorrido	Mostrar menú, selección, tutoriales, actividades y resultados.	C3_01, C3_02, C3_03, C3_05 a C3_10, C3_12 y C3_13.
Herramientas y organización	Confirmar versión, carpetas y recursos del proyecto.	C4_01; C4_02_1, C4_02_2, C4_02_3 y C4_02_17.
Jerarquías y escenas	Mostrar objetos del menú, selector, minijuego y escenas.	C4_04, C4_05, C4_06 y C4_07.
Sistemas y mecánicas	Mostrar eventos, estrellas, secuencia y controlador de voz.	C4_08, C4_09, C4_10 y C4_11.
Consola	Registrar el estado de la prueba de voz mostrada.	C4_13: 0 errores y 1 advertencia visible.
Compilación	Mostrar plataforma e índices de escenas.	C4_03; la captura de resultado se difiere hasta eliminar la advertencia.

Nota. Elaboración propia a partir del proyecto y sus registros técnicos.

La versión documentada integra recursos gráficos 2D, seis escenas, cuatro mecánicas, retroalimentación audiovisual, progreso local y lectura apoyada por voz. La separación entre escenas, controladores, repositorios y servicios permite localizar las responsabilidades necesarias para mantenimiento.

La valoración del cierre se remite al apartado 4.2.11 y a la Tabla 22. Los registros disponibles documentan el comportamiento técnico de navegación, mecánicas, voz, persistencia y compilación en la versión revisada. En conjunto, muestran cómo los sistemas compartidos se integran con los cuatro mundos y cómo el resultado de cada actividad se incorpora al flujo general, sin extender la conclusión a condiciones no registradas.

Las evidencias de este manual son técnicas. Las decisiones visuales y de interacción no equivalen a resultados validados con niños; la usabilidad, el rendimiento y el posible valor educativo requieren evaluaciones posteriores.

Conclusiones

Las conclusiones que se presentan a continuación responden a los objetivos específicos planteados en el Capítulo I y se apoyan en la evidencia documentada en los Capítulos III y IV.

En relación con la creación de los recursos gráficos, se integraron personajes, fondos, objetos, interfaces y recursos 2D para los cuatro mundos temáticos. La organización por carpetas y funciones favorece la reutilización dentro de Unity. Las ilustraciones del manual documentan la referencia conceptual, la reconstrucción vectorial, las paletas, las poses, la preparación de los PNG y su incorporación al proyecto.

El diseño de la estructura de escenas, mundos y niveles consolidó un recorrido desde el menú principal hasta la selección, los tutoriales, las actividades y el retorno al selector. La centralización de los nombres de escena y las transiciones contribuye al mantenimiento del flujo. Las evidencias registradas documentan la carga de escenas, el retorno y la actualización del progreso dentro de la versión comprobada.

La programación de las mecánicas permitió implementar clic, selección directa, validaciones y retroalimentación en las actividades de música, lectura, tamaños y emociones. Los controladores reciben la entrada, aplican las reglas de cada actividad, emiten eventos de resultado y enlazan el cierre con los sistemas de estrellas y progreso. Esta separación permite seguir el flujo de ejecución sin depender únicamente de las tablas de clases.

Mediante programación se implementaron los sistemas complementarios de progreso, estrellas, activación visual de planetas, audio y lectura apoyada por reconocimiento de voz. PlayerPrefs conserva el avance local y los repositorios separan

parte de la lógica de persistencia. El módulo de voz depende de Windows, del micrófono y de sus permisos, por lo que su funcionamiento no puede generalizarse fuera del entorno documentado.

En la versión y los registros documentados, se aprobaron 33 de 33 casos EditMode y 8 de 8 casos PlayMode, sin fallos ni omisiones. La comprobación del módulo de voz presentó cero errores y una advertencia visible, y la compilación para Windows fue registrada como exitosa. Estos resultados permiten afirmar que los casos técnicos ejecutados cumplieron sus resultados esperados bajo las condiciones documentadas; no demuestran ausencia absoluta de defectos ni permiten concluir usabilidad, comprensión, aceptación, aprendizaje o efectos cognitivos en niños.

Recomendaciones

En primer lugar, se recomienda que la entrega final conserve una única campaña de pruebas EditMode y PlayMode vinculada con la misma versión del ejecutable y del código fuente. También conviene archivar la captura de la consola, la evidencia del módulo de voz y el registro de compilación junto con la fecha y la versión, de modo que cada afirmación técnica pueda relacionarse con un resultado reproducible.

Se recomienda documentar el rendimiento con Unity Profiler en los equipos objetivo. El registro debe incluir FPS, tiempo por fotograma, memoria, CPU/GPU, tiempos de carga, resolución y configuración del hardware. Estos datos permitirán establecer criterios de aceptación y distinguir una comprobación visual de una evaluación cuantitativa de rendimiento.

El módulo de voz debe evaluarse mediante un protocolo reproducible en al menos dos equipos. Conviene registrar idioma, micrófono, distancia, ruido, frases, variantes, permisos, tiempo de espera, transcripción y alternativa de reintento. La documentación debe informar su dependencia de Windows y evitar cualquier afirmación de confiabilidad que no esté respaldada por resultados.

El manual debe probarse con una persona adulta que no haya participado en el desarrollo y diferenciar preparación, supervisión y acciones del niño. Una evaluación posterior con el público objetivo requerirá revisión ética, consentimiento de representantes e instrumentos de usabilidad adecuados. Hasta entonces, la claridad y la adecuación por edad deben presentarse como decisiones de diseño.

La entrega final debe conservar el ejecutable, su versión y las evidencias técnicas asociadas.

Trabajos Futuros

Como trabajo inmediato se plantea repetir la campaña técnica sobre la versión definitiva y conservar sus evidencias en una carpeta identificable. Más adelante puede ampliarse la comprobación con mediciones de rendimiento mediante Unity Profiler, pruebas de regresión, evaluación del manual y revisión del módulo de voz bajo distintas condiciones ambientales.

En una fase posterior, el prototipo puede incorporar nuevos mundos, más actividades y una alternativa de reconocimiento de voz menos dependiente del sistema operativo. La evaluación con docentes, cuidadores y niños deberá diseñarse como un estudio separado, con criterios éticos y métricas de usabilidad, comprensión y experiencia; solo entonces podrán discutirse resultados relacionados con el público objetivo.

Referencias

[1] United Nations Children's Fund, *Early Childhood Development: UNICEF Vision for Every Child*. United Nations, 2023. doi: 10.18356/9789213585429.

[2] *Global report on early childhood care and education: the right to a strong foundation*. UNESCO, 2024. doi: 10.54675/FWQA2113.

[3] L. C. Rodríguez Timaná, J. F. Castillo García, T. Bastos Filho, A. A. Ocampo González, N. R. Hincapié Monsalve, y N. J. Valencia Jimenez, «Use of Serious Games in Interventions of Executive Functions in Neurodiverse Children: Systematic Review», *JMIR Serious Games*, vol. 12, p. e59053, dic. 2024, doi: 10.2196/59053.

[4] M. S. Alotaibi, «Game-based learning in early childhood education: a systematic review and meta-analysis», *Front. Psychol.*, vol. 15, p. 1307881, abr. 2024, doi: 10.3389/fpsyg.2024.1307881.

[5] C. Chu, L. Paatsch, L. Kervin, y S. Edwards, «Digital play in the early years: A systematic review», *Int. J. Child-Comput. Interact.*, vol. 40, p. 100652, jun. 2024, doi: 10.1016/j.ijcci.2024.100652.

[6] Z.-M. Liu, C.-Q. Chen, X.-L. Fan, C.-C. Lin, y X.-D. Ye, «Usability and Effects of a Combined Physical and Cognitive Intervention Based on Active Video Games for Preschool Children», *Int. J. Environ. Res. Public. Health*, vol. 19, n.º 12, p. 7420, jun. 2022, doi: 10.3390/ijerph19127420.

[7] Kilis 7 Aralık University, Kilis, 79000, Turkey, H. Güler, Y. Şahinkayasi, Hatay Mustafa Kemal University, Hatay, 31000, Turkey, H. Şahinkayasi, y Hatay Mustafa Kemal University, Hatay, 31000, Turkey, «Design and development process of a digital game to acquire preschoolers' self-care skills: a model practice», *Afr. Educ. Res. J.*, vol. 11, n.º 3, pp. 269-292, 2023, doi: 10.30918/AERJ.113.23.030.

[8] J. A. Widjaja, L. Jefferson, M. F. B. Siahaan, y A. Chow, «Utilizing Game Development Life Cycle Method to Develop an Educational Game for Basic Mathematics Using Unity 2D Game Engine», *IJISIT Int. J. Comput. Sci. Inf. Technol.*, vol. 1, n.º 1, pp. 20-30, jun. 2024, doi: 10.55123/ijisit.v1i1.6.

[9] Y. P. Defaz Gallardo, E. A. Bustillos Velez, y L. S. Defaz Gallardo, «Guía de Gimnasia Cerebral para fortalecer el desarrollo cognitivo de los niños de educación inicial», *Rev. Criterio*, vol. 1, n.º 1, pp. 18-29, oct. 2021, doi: 10.62319/criterio.v.1i1.2.

[10] A. Masa y J. Tillaguango, «Gimnasia Cerebral, método de estimulación para el Desarrollo Psicomotor de niños de 4 a 5 años: Brain Gymnastics, stimulation method for the Psychomotor Development of children from 4 to 5 years old», *LATAM Rev. Latinoam. Cienc. Soc. Humanidades*, vol. 4, n.º 2, may 2023, doi: 10.56712/latam.v4i2.659.

[11] C. A. Aquieta Guamán, «Implementación de Aplicaciones Educativas enfocadas en el desarrollo cognitivo de los niños de inicial II del Centro Infantil Caritas Felices», Tesis de maestría, Instituto Superior Tecnológico Universitario Rumiñahui, Sangolquí, Ecuador, 2024. [En línea]. Disponible en: https://repositorio.ister.edu.ec/handle/68000/340

[12] R. A. Estrada Zamora, M. P. Yanza Paguay, M. E. Kliger Bone, y M. J. Muñoz Zamora, «Impacto de la Tecnología en el Desarrollo Cognitivo de Niños preescolares: Integración en el Aprendizaje», *Mediciencias UTA*, vol. 8, n.º 4, pp. 54-68, oct. 2024, doi: 10.31243/mdc.uta.v8i4.2626.2024.

[13] K. M. Solano García y M. P. Piñaloza Martínez, «Herramientas tecnológicas y desarrollo cognitivo en el proceso educativo en los niños de educación inicial, subnivel II, Santa Rosa 2023», Trabajo de titulación, Universidad Técnica de Machala, Machala, Ecuador, 2024. [En línea]. Disponible en: https://repositorio.utmachala.edu.ec/handle/48000/22878

[14] A. Hämäläinen *et al.*, «A multimodal educational game for 3-10-year-old children: collecting and automatically recognising European Portuguese children's speech», en *Speech and Language Technology in Education (SLaTE 2013)*, ISCA, ago. 2013, pp. 31-36. doi: 10.21437/SLaTE.2013-5.

[15] S. Dutta, D. Irvin, J. Buzhardt, y J. H. L. Hansen, «Activity focused Speech Recognition of Preschool Children in Early Childhood Classrooms», en *Proceedings of the 17th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2022)*, Seattle, Washington: Association for Computational Linguistics, 2022, pp. 92-100. doi: 10.18653/v1/2022.bea-1.13.

[16] H. Jung, H. J. Kim, S. So, J. Kim, y C. Oh, «TurtleTalk: An Educational Programming Game for Children with Voice User Interface», en *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*, Glasgow Scotland Uk: ACM, may 2019, pp. 1-6. doi: 10.1145/3290607.3312773.

[17] K. Junttila *et al.*, «Gaming enhances learning-induced plastic changes in the brain», *Brain Lang.*, vol. 230, p. 105124, jul. 2022, doi: 10.1016/j.bandl.2022.105124.

[18] S. H. Nustad y L. M. Abrahamsson, «Passive and active screen time relate differently to attention in preschool children», *Front. Psychol.*, vol. 17, p. 1737937, mar. 2026, doi: 10.3389/fpsyg.2026.1737937.
[19] N. Lakicevic, M. Manojlovic, E. Chichinina, P. Drid, y Y. Zinchenko, «Screen time exposure and executive functions in preschool children», *Sci. Rep.*, vol. 15, n.º 1, p. 1839, ene. 2025, doi: 10.1038/s41598-024-79290-6.
[20] C. M. D. L. Martins *et al.*, «A Network Perspective on the Relationship between Screen Time, Executive Function, and Fundamental Motor Skills among Preschoolers», *Int. J. Environ. Res. Public. Health*, vol. 17, n.º 23, p. 8861, nov. 2020, doi: 10.3390/ijerph17238861.
[21] K. Xu, S. Geng, D. Dou, y X. Liu, «Relations between Video Game Engagement and Social Development in Children: The Mediating Role of Executive Function and Age-Related Moderation», *Behav. Sci.*, vol. 13, n.º 10, p. 833, oct. 2023, doi: 10.3390/bs13100833.
[22] C. Roma, «Relación entre videojuegos y funciones ejecutivas en niños de nivel primario: una revisión sistemática», *JAIIO Jorn. Argent. Informática*, vol. 10, n.º 4, pp. 83-101, 2024.
[23] R. S. Calabrò y A. Quartarone, «Neurorehabilitation as Network Perturbation: Shaping Neuroplasticity with Robotics, Virtual Reality, and Neuromodulation», *Biomedicines*, vol. 14, n.º 2, p. 411, feb. 2026, doi: 10.3390/biomedicines14020411.
[24] K. Pradeep, R. Sulur Anbalagan, A. P. Thangavelu, S. Aswathy, V. G. Jisha, y V. S. Vaisakhi, «Neuroeducation: understanding neural dynamics in learning and teaching», *Front. Educ.*, vol. 9, p. 1437418, dic. 2024, doi: 10.3389/feduc.2024.1437418.
[25] X. Shi y Y. Yu, «Design and application of art creation education system based on generative adversarial network», *Discov. Artif. Intell.*, vol. 6, n.º 1, p. 24, dic. 2025, doi: 10.1007/s44163-025-00682-2.
[26] M. J. Quiroz-Vélez y M. E. Vaca-Cárdenas, «La gimnasia cerebral para fortalecer los aprendizajes significativos en la Unidad Educativa Cruz del Norte», *Polo Conoc.*, vol. 8, n.º 6, pp. 311-329, 2023.
[27] M. E. Tonato Guamangallo y J. S. Torres Peñafiel, «Gimnasia cerebral en la atención de los niños», *Polo Conoc.*, vol. 8, n.º 3, pp. 142-154, 2023.
[28] K. Schwaber y J. Sutherland, «The Scrum Guide», nov. 2020. Accedido: 1 de septiembre de 2026. [En línea]. Disponible en: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf
[29] J. Coleman *et al.*, «The Kanban Guide», may 2025. Accedido: 1 de septiembre de 2026. [En línea]. Disponible en: https://kanbanguides.org/the-kanban-guide/2025.5/pdf/kanban-guide.v2025.5.en.pdf
[30] Unity Technologies, «2D game creation workflow», Unity Manual 6000.1. Accedido: 1 de septiembre de 2026. [En línea]. Disponible en: https://docs.unity3d.com/6000.1/Documentation/Manual/2d-game-creation-wokflow.html
[31] Unity Technologies, «Introduction to programming in Unity», Unity Manual 6000.0. Accedido: 1 de septiembre de 2026. [En línea]. Disponible en: https://docs.unity3d.com/6000.0/Documentation/Manual/intro-to-scripting.html
[32] Godot Engine, «Introduction to 2D», Godot Engine documentation. Accedido: 1 de septiembre de 2026. [En línea]. Disponible en: https://docs.godotengine.org/en/stable/tutorials/2d/introduction_to_2d.html
[33] Godot Engine, «Learn to code with GDScript», Godot Engine documentation. Accedido: 1 de septiembre de 2026. [En línea]. Disponible en: https://docs.godotengine.org/en/stable/getting_started/introduction/learn_to_code_with_gdscript.html
[34] Epic Games, «Paper 2D Overview in Unreal Engine», Unreal Engine documentation. Accedido: 1 de septiembre de 2026. [En línea]. Disponible en: https://dev.epicgames.com/documentation/unreal-engine/paper-2d-overview-in-unreal-engine?lang=en-US
[35] Epic Games, «Coding in UE: Blueprint vs. C++», Unreal Engine documentation. Accedido: 1 de septiembre de 2026. [En línea]. Disponible en: https://dev.epicgames.com/documentation/en-us/unreal-engine/coding-in-unreal-engine-blueprint-vs-cplusplus
[36] Unity Technologies, «Integrated development environment (IDE) support», Unity Manual 6000.0. Accedido: 1 de septiembre de 2026. [En línea]. Disponible en: https://docs.unity3d.com/6000.0/Documentation/Manual/scripting-ide-support.html
[37] Microsoft, «Visual Studio Tools for Unity», Microsoft Learn. Accedido: 1 de septiembre de 2026. [En línea]. Disponible en: https://learn.microsoft.com/en-us/visualstudio/gamedev/unity/get-started/visual-studio-tools-for-unity
[38] JetBrains, «Game development for Unity», JetBrains Rider Documentation. Accedido: 1 de septiembre de 2026. [En línea]. Disponible en: https://www.jetbrains.com/help/rider/Unity.html
[39] Microsoft, «N-tier architecture style», Azure Architecture Center. Accedido: 1 de septiembre de 2026. [En línea]. Disponible en: https://learn.microsoft.com/en-us/azure/architecture/guide/architecture-styles/n-tier

[40] S. Smith, «Overview of ASP.NET Core MVC», Microsoft Learn. Accedido: 1 de septiembre de 2026. [En línea]. Disponible en: https://learn.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-10.0