PROYECTO FIN DE GRADO

TÍTULO: Ampliación de un videojuego terapéutico para mejora del procesamiento auditivo: creación de un módulo de escucha dicótica

AUTOR: Pablo Lucas Olivares

TITULACIÓN: Grado en Ingeniería de Sonido e Imagen

TUTORA: Martina Eckert

DEPARTAMENTO: Ingeniería Audiovisual y Comunicaciones

VºBº TUTORA

Miembros del Tribunal Calificador:

PRESIDENTE: Sergio López Gregorio

TUTORA: Martina Eckert

SECRETARIO: Enrique Rendón Angulo

Fecha de lectura: 26 de julio de 2023

Calificación:

El Secretario,

Resumen

Este proyecto de fin de grado tiene como objetivo la ampliación del videojuego terapéutico en desarrollo "El Planeta Sonoa" para niños con problemas neurológicos relacionados con la audición, la atención y el lenguaje. La finalidad general del videojuego es transformar las diferentes pruebas y ejercicios que deben realizar los pacientes en juegos mucho más atractivos y entretenidos para ellos, pero sin perder su finalidad terapéutica.

La ampliación realizada en este proyecto consiste en la creación de un nuevo módulo del juego para el entrenamiento de la escucha dicótica (escucha de diferentes estímulos en cada oído) en los pacientes.

El módulo, llamado "Tropikus", tiene una ambientación tropical y de playa, con la finalidad de resultar llamativa a los niños. En el juego, debes ayudar a dos guacamayos a abrir cofres para obtener objetos. Los guacamayos dicen diferentes palabras que el paciente debe escuchar e identificar sin ayudas visuales para posteriormente buscar en la isla dibujos que se correspondan con ellas y excavar para obtener las recompensas. El juego debe ser totalmente ajustable a las necesidades de los pacientes. Estos ajustes, que se corresponden con el número de sílabas de las palabras escuchadas, el ruido de fondo, el número de dibujos que aparecen en la isla y el número de rondas que se jugarán, se realizan mediante la página web de "El Planeta Sonoa" y son totalmente invisibles al jugador. Paralelamente, se crea también un menú de ajuste que se activa si no es posible jugar en red.

En el momento del desarrollo de este proyecto el videojuego "El Planeta Sonoa" consta con dos módulos ya implementados para otros tipos de ejercicios y se desarrollan simultáneamente a este otros tres más. Por tanto, se trata de un trabajo en constante comunicación entre los miembros del equipo y con la necesidad de unificar su funcionamiento y planteamiento. El proyecto, además, se desarrolla en colaboración con profesionales de la audiología que aportan información sobre las necesidades reales que tienen sus pacientes para que los juegos se ajusten a ellas.

Abstract

This final degree project aims to expand the therapeutic videogame “El Planeta Sonoa” for children with neurological problems related to hearing, attention and language. The overall purpose of the videogame is to transform the various tests and exercises that patients must perform into much more appealing and entertaining games for them, while maintaining their therapeutic purpose.

The expansion carried out in this project consists of creating a new module of the game for training dichotic listening (listening to different stimuli in each ear) in patients.

The module, called “Tropikus”, has a tropical and beach setting, designed to be attractive to children. In the game, you must help two macaws open chests to obtain objects. The macaws say different words that the patient must listen to and identify without visual aids, and then search the island for drawings that correspond to those words and dig to obtain rewards. The game must be fully adjustable to the needs of the patients. These adjustments, which correspond to the number of syllables in the heard words, the background noise, the number of drawings that appear on the island, and the number of rounds to be played, are made through the website of “El Planeta Sonoa” and are completely invisible to the player. Additionally, a settings menu is also created that is activated if it is not possible to play online.

At the time of the development of this project, the video game “El Planeta Sonoa” already has two modules implemented for other types of exercises, and three more are being developed simultaneously. Therefore, it is a work that requires constant communication among team members and the need to unify its operation and approach. The project also involves collaboration with audiology professionals who provide information about the real needs of their patients so that the games can be adjusted to them.

Acrónimos

AEVI: Asociación Española del Videojuego

DPAC: Desorden del Procesamiento Auditivo Central

SNAC: Sistema Nervioso Auditivo Central

GAMMA: Grupo de investigación de Aplicaciones Multimedia y Acústica

CITSEM: Centro de Investigación en Tecnologías Software y Sistemas Multimedia para la Sostenibilidad

BEPADI: Batería de Evaluación del Procesamiento Auditivo Dicótico

GDD: *Game Design Document*

NPC: *Non-Playable Character*

HUD: *Heads-Up Display*

ODS: Objetivo de Desarrollo Sostenible

DAW: *Digital Audio Workstation*

ID: Identificador

Índice de contenido

Índice de figuras

Índice de tablas

1. Introducción

En la actualidad los videojuegos tienen un gran impacto cultural y social. Según el informe anual de la AEVI (Asociación Española del Videojuego) [1], en España la base de jugadores aumenta cada año, con un 77% de niños entre 6 y 14 años que juegan a videojuegos semanalmente. Esto ha propiciado la aparición de muchos subgéneros de videojuegos con finalidades muy diversas como la competición, el arte o la educación. Uno de estos subgéneros son los videojuegos serios y terapéuticos. En la actualidad, los videojuegos terapéuticos han adquirido una importancia significativa en el campo de la salud y el bienestar. Estos juegos están diseñados específicamente para abordar diferentes condiciones médicas y psicológicas, y proponer nuevas formas de tratamiento y ejercicio.

La importancia de los videojuegos terapéuticos aumenta al enfocarse en los niños. Los niños de la actualidad han crecido en contacto con estas tecnologías desde su nacimiento, apareciendo así el término "nativos digitales" [2]. Esto supone que, para un niño, una herramienta como un videojuego puede resultar más accesible y familiar que otras.

Es importante destacar que los videojuegos terapéuticos no buscan reemplazar a los tratamientos convencionales, sino que se utilizan como una herramienta complementaria. Estos juegos pueden ser utilizados en combinación con terapia tradicional, medicación u otras intervenciones médicas. El objetivo de los videojuegos terapéuticos es complementar las terapias y tratamientos a través de la gamificación [3], aplicar elementos y técnicas propias de los juegos en contextos no relacionados con ellos, lo cual puede resultar muy atractivo cuando los pacientes son niños.

Debido, precisamente, al auge de los videojuegos en la sociedad, se ha propiciado también la creación de cada vez más herramientas de libre acceso para su creación, acercando a los usuarios la capacidad de crear videojuegos que antes no estaban a su alcance. Herramientas como el motor de videojuegos Unity [4] o Blender [5], software de modelado y animación de objetos 3D, son gratuitas y accesibles para cualquier usuario.

Como resultado de la combinación de estos elementos surge el videojuego del que forma parte este proyecto, "El Planeta Sonoa", del que se ha desarrollado un nuevo módulo para el entrenamiento de la escucha dicótica detallado en esta memoria.

2. Antecedentes

El GAMMA (grupo de investigación de Aplicaciones MultiMedia y Acústica) forma parte del CITSEM (Centro de Investigación en Tecnologías Software y Sistemas Multimedia para la Sostenibilidad), con sede en la Escuela de Ingeniería y Sistemas de Telecomunicación de la Universidad Politécnica de Madrid. Este grupo tiene varias líneas de trabajo, siendo una de ellas el diseño y desarrollo de juegos serios para su aplicación en entornos educativos.

Entre los proyectos de videojuegos terapéuticos en desarrollo por este grupo de investigación se encuentra *Phiby's Adventures 3D* [6], enfocado a niños con movilidad reducida haciendo uso del dispositivo de captación de movimientos Kinnect, cuyo desarrollo comenzó en 2018. En el mismo año se inicia también el desarrollo del primer módulo que conforma el proyecto de "El Planeta Sonoa", que tiene como objetivo el entrenamiento de capacidades neurológicas que puedan verse afectadas en las personas diagnosticadas con DPAC (desorden del procesamiento auditivo central).

2.1 Desorden del procesamiento auditivo central (DPAC)

El DPAC es un desorden relacionado con las operaciones neurobiológicas que tienen lugar dentro del sistema auditivo. Las personas que sufren este tipo de desorden no tienen problemas para percibir un sonido como los pacientes que sufren dolencias como la hipoacusia (conocida comúnmente como sordera), pero sí presentan otras dificultades auditivas a la hora de comprender, interpretar y recordar lo escuchado. Como consecuencia de esto, a lo largo de los años se ha diagnosticado a quienes sufren de DPAC con otros trastornos como el déficit de atención. En su artículo, Óscar Cañete [7] expone los síntomas identificados en personas con DPAC (Tabla 1).

Tabla 1: Habilidades auditivas importantes en el proceso de aprendizaje, por Óscar Cañete

Discriminación	Diferenciación de sonidos de diferente frecuencia, duración o intensidad.
Localización	Ubicación de la fuente sonora.
Discriminación auditiva	Discriminación de los elementos fonémicos del habla que son acústicamente similares.
Encierro auditivo	Comprensión de un mensaje o palabra "completo" cuando una porción de este está ausente.
Separación auditiva en ruido	Identificación del hablante primario en presencia de ruido de fondo.
Asociación auditiva	Capacidad para otorgar un significado a las palabras.
Memoria auditiva	Capacidad para almacenar y recordar un estímulo en orden o secuencia apropiada.
Atención auditiva	Capacidad para dirigir y mantener la atención hacia una señal acústica relevante por un período apropiado de tiempo.

Este tipo de desórdenes aparecen en diversas poblaciones de pacientes, como aquellos con una patología en el SNAC (sistema nervioso auditivo central), un desorden del desarrollo neural u otros tipos de deterioro del sistema nervioso.

Los pacientes con DPAC pueden ver afectado el normal desarrollo del lenguaje, con el impacto que esto supone en el éxito académico o la efectividad comunicativa. Existen diferentes ejercicios y tratamientos para mejorar las habilidades auditivas en estos pacientes. Crear nuevas formas de realizar estos ejercicios es la finalidad con la que nace este proyecto, "El Planeta Sonoa".

2.2 "El Planeta Sonoa"

Este proyecto está dirigido por la profesora Martina Eckert, miembro del GAMMA. Consiste en un videojuego terapéutico titulado inicialmente "El planeta de Amalia" y renombrado a "El Planeta Sonoa" durante el desarrollo de este proyecto y en consenso con los otros estudiantes que desarrollaron módulos para este juego en paralelo. Este juego, destinado a niños entre 5 y 14 años con problemas neurológicos relacionados con el DPAC, tiene la finalidad de brindar herramientas para que los terapeutas realicen sus diferentes pruebas para el tratamiento, entrenamiento y diagnóstico de una forma divertida y motivadora para los niños, mediante la gamificación de las distintas pruebas y ejercicios necesarios para ello.

Se trata de una colección de juegos destinados a realizar diferentes ejercicios. El videojuego se realiza en colaboración con terapeutas del Centro de Audiología y Logopedia Aurea TAV Madrid [8], especialistas en audición y comunicación, que se encargan de especificar las necesidades y características que deben tener los diferentes juegos incluidos en "El Planeta Sonoa", así como indicar los resultados que se necesitan registrar para el estudio de los pacientes. El juego es, además, ajustable a través de una página web asociada al juego que ya está implementada, para adecuarlo a las necesidades de cada paciente y monitorizar los resultados obtenidos.

La creación de módulos para el videojuego se ha extendido a lo largo de los años, por lo que cuenta con proyectos realizados por alumnos y miembros del GAMMA desde 2018. Concretamente, consta de dos módulos de juego ya implementados:

- "Polaris", desarrollado por Luna Sampedro Jiménez [9] y mejorado por Esther García Medina [10]. Módulo para entrenar la memoria auditiva y la atención. Se ambienta en un entorno glaciar en el que se debe repetir una cadena de sonidos emitidos por un pingüino mientras otros producen ruidos.
- "Selvalia", desarrollado por Cristina Ojeda Egocheaga [11]. Módulo para entrenar la discriminación de elementos fonémicos del habla acústicamente similares y la asociación auditiva. Se ambienta en una selva en la que el jugador debe ayudar a una familia de monos.

Además, el videojuego cuenta con un menú central navegable para escoger el ejercicio que realizar, que dispone de un sistema de gestión de usuarios que se conecta con una base de datos en la que se registran los parámetros de configuración y los resultados de cada jugador. Las labores de implementación de la base de datos y la web, la integración de los módulos en el juego global en Unity y la adecuación de los módulos han sido llevadas a cabo por Miguel Jiménez Arévalo, técnico de laboratorio del grupo GAMMA.

En el tiempo en que se desarrolla el módulo objetivo de este proyecto se realizan simultáneamente cuatro nuevos módulos, conformando un equipo en el que cada integrante se centra en el desarrollo de uno de ellos, pero pudiendo aportar opiniones al resto de trabajos con el fin de unificar el formato de los módulos creados y facilitar las posteriores labores de integración en el videojuego global. Los cuatro nuevos módulos desarrollados en este tiempo son:

- "Cavernia", desarrollado por Bogdan Morar [12]. Módulo para entrenar el entendimiento de historias cortas escuchadas. Se ambienta en un bosque, con una cueva con osos que narran las historias al jugador.
- "Kobarag", desarrollado por Mario Ortega García [13]. Módulo para entrenar la memoria auditiva, con cadenas de tonos largos y cortos o graves y agudos. Se ambienta en un desierto, con un mercado en el que unas cobras reproducen las diferentes señales que el jugador debe repetir.
- "Pandalia", en desarrollo por Moheng Li [14]. Módulo para entrenar la capacidad de detección de sonidos en el espacio 3D (mediante el uso de gafas de realidad virtual). Se ambienta en un poblado asiático, en el que unos pandas necesitan ayuda para colocar farolillos en la celebración del Año Nuevo. Estos farolillos deben ser identificados espacialmente mediante sonidos.
- "Tropikus", módulo en el que se centra este proyecto. Su objetivo es el entrenamiento de la escucha dicótica. Se ambienta en una isla tropical en la que una pareja de guacamayos recita dos palabras que el jugador debe buscar en la playa.

En total, el juego completo contiene seis módulos que se identifican con una isla o continente del planeta del juego base y que comparten características similares como el ajuste del ruido de fondo y la selección de niveles por parte del terapeuta.

2.3 Escucha dicótica

La escucha dicótica es un fenómeno perceptual que ocurre cuando se presentan estímulos auditivos diferentes en cada oído al mismo tiempo. Es un proceso que nos permite experimentar y analizar cómo nuestro cerebro procesa y organiza la información auditiva proveniente de distintas fuentes.

La escucha dicótica también se ha utilizado en estudios clínicos para evaluar el funcionamiento auditivo y detectar posibles déficits en el procesamiento auditivo central. Por ejemplo, en personas con trastornos del lenguaje o dificultades de atención, se pueden observar patrones atípicos de procesamiento en la escucha dicótica.

Cuando se realiza un experimento de escucha dicótica, se utilizan auriculares o altavoces separados para enviar diferentes estímulos a cada oído. En función del estudio realizado se le puede pedir al participante prestar atención a uno o ambos oídos y luego se le solicita que informe lo que escuchó en cada caso. En el campo de la audiología se denomina como BEPADI (Batería de Evaluación del Procesamiento Auditivo Dicótico) [15] a la evaluación que se lleva a cabo para identificar posibles dificultades en el procesamiento de la información auditiva, específicamente en relación con la integración sensorial auditiva. Se busca comprender si existen alteraciones funcionales en lugar de atribuir los problemas a la audición misma o a causas periféricas relacionadas con ella.

En el caso de este videojuego, que también debe jugarse con auriculares para obtener una escucha dicótica real, el jugador escuchará una palabra en cada oído. Según las especificaciones detalladas por las audiólogas del centro Aurea TAV, ambas palabras deben contener el mismo número de sílabas (en el juego es ajustable a una, dos y hasta tres sílabas), siendo mayor la dificultad cuanto menor sea el número de sílabas, debido a que se escucha menos información que permita reconocer la palabra escuchada.

Al ser un fenómeno estrechamente relacionado con la comprensión del lenguaje y que ha sido principalmente estudiado en inglés, existen pocos recursos para el estudio de la escucha dicótica en castellano. Es por esta razón que el sencillo hecho de disponer de una nueva colección de palabras pregrabadas como las pertenecientes a este videojuego, junto a la posibilidad de escucharlas simultáneamente en cada oído y de forma aleatoria ya supone un material de trabajo muy interesante.

La escucha dicótica ha sido ampliamente estudiada en el campo de la psicología y la neurociencia para comprender cómo el cerebro procesa la información auditiva y cómo se lleva a cabo la atención selectiva. Uno de los hallazgos más importantes es el efecto de la lateralización hemisférica, donde el oído derecho se encuentra más conectado con el hemisferio cerebral izquierdo y viceversa. Este fenómeno se conoce como “ventaja del oído derecho”, que produce un mayor rendimiento en este oído debido al cruzamiento de las vías. Como resultado de este efecto se deduce que el tiempo de procesamiento de lo escuchado en el oído izquierdo es más lento, haciendo en ocasiones necesario el uso de retardos en uno de los oídos para ajustar el tiempo de reproducción.

3. Objetivos y requisitos del proyecto

Al tratarse de un proyecto desarrollado en colaboración con diferentes compañeros, los objetivos del desarrollo de este módulo perteneciente a "El Planeta Sonoa" responden a objetivos individuales y objetivos relacionados con la integración del juego en el juego global. Los objetivos del proyecto son:

1. Implementación de una mecánica de juego para el entrenamiento de la escucha dicótica, en la que el jugador escuche una palabra en cada oído y deba discernir qué palabras ha escuchado. El juego debe ser ajustable a tres niveles de dificultad: palabras monosílabas, bisílabas y trisílabas, pudiendo además seleccionar hasta un total de cinco niveles con diferentes configuraciones de número de opciones, número de rondas, retardo y ruidos de fondo. Para mejorar su accesibilidad y facilitar su uso, el juego debe poderse jugar con teclado y ratón o con un mando de videoconsola compatible. Además, se busca un rendimiento eficiente en todo tipo de equipos, para evitar que esto sea un impedimento en su uso.

2. Implementación de una estética y elementos de juego motivadores para niños, lo cual añade el requisito de no mostrar elementos violentos como armas, bestias o la muerte del jugador. Creación de diferentes elementos gráficos para la interfaz de usuario, así como modelos 3D y texturas, haciendo uso también de diferentes *assets* (recursos) y modelos externos con licencias de uso. Ya que se cuenta con el requisito de obtener un buen rendimiento del juego, los modelos utilizados serán de tipo *low-poly*, es decir, creados con pocos polígonos, más ligeros computacionalmente.

3. Implementación de la posibilidad de configuración de parámetros de dificultad de juego y ajustes de sonidos y ruido por los terapeutas, además de registrar los resultados para poder facilitar su envío a la base de datos. Se deben poder ajustar los niveles de los de los sonidos ambientes y ruidos distractorios, además de poder cambiar el lado del que suena cada ruido. El objetivo es que el terapeuta realice las configuraciones personalizadas para cada jugador registrado desde la interfaz web, sin embargo, se debe implementar también la posibilidad de efectuar los mismos ajustes a través de un menú propio del juego, para fines demostrativos o para posibilitar el uso del juego sin conexión a internet. Durante la elaboración de este proyecto, el técnico de laboratorio va a llevar a cabo las ampliaciones necesarias en la base de datos para los nuevos módulos. Una vez el módulo haya sido implementado en el juego global, los resultados registrados se deben almacenar por paciente y ser enviados a la base de datos. De esta manera, en la interfaz web pueden mostrarse como forma de tablas y gráficas para obtener información sobre las dificultades que se han tenido en el proceso de juego.

4. Desarrollo de cinemáticas explicativas del juego, para crear una experiencia más inmersiva. Deben ser sencillas y suficientes para entender el procedimiento y la finalidad del juego y aportar información al jugador, con la opción de omitirse para el caso de la introducción. Añadir también otras ayudas visuales para el entendimiento del juego como mensajes explicativos de los controles para realizar las diferentes acciones en el juego como la interacción, el movimiento o la cámara.

5. Creación de documentación explicativa y comentarios del código del juego, con la finalidad de que puedan ser consultados en el futuro para posibles cambios, mejoras o expansiones. Entre la documentación que debe contener el proyecto se incluye un manual de juego (Anexo I) y un GDD (*Game Design Document)* que se crea posteriormente al desarrollo del juego. Como prioridad, se pretende utilizar el inglés en la redacción del código fuente y sus comentarios para hacerlo más accesible.

4. Solución propuesta

En base a los objetivos y requisitos planteados, se busca una solución adecuada que cumpla con todos ellos y pueda desarrollarse en el tiempo de trabajo disponible. En el caso pertinente a este proyecto de fin de grado, centrado en el desarrollo del apartado de "Tropikus" de "El Planeta Sonoa", se ha optado por un método de Ruta Crítica [16]. La razón por la que se escoge este modelo es la necesidad de dividir el trabajo en pequeñas tareas, para después priorizarlas en función de su importancia e impacto en el proyecto. De esta forma, es posible visualizar qué tareas es necesario finalizar en primer lugar para poder continuar con otras que dependan de ella, y qué tareas pueden tener menos prioridad al no inhabilitar el desarrollo del proyecto global. El método de Ruta Crítica implica, además, estimar el tiempo necesario para finalizar cada una de las tareas para tenerlo en cuenta a la hora de priorizar cuál realizar en siguiente lugar. Sin embargo, prevé la imprecisión en estas estimaciones debido a la aparición de nuevos requisitos o limitaciones.

El desarrollo comienza con el análisis del estado actual del videojuego y de los requisitos especificados para el módulo que atañe a este proyecto. Debido a que el proyecto parte de un juego como base cuya temática es un planeta con animales en diferentes hábitats, se ha elegido una temática de una pequeña isla tropical, eligiendo por esta razón "Tropikus" como nombre del módulo. En esta isla habita una pareja de guacamayos (una especie de loros) que necesitan la ayuda del jugador. Se escoge este animal debido a su capacidad de hablar repitiendo palabras y su presencia en hábitats tropicales. Además, se trata de una pareja para justificar la escucha de dos estímulos en cada oído para generar la escucha dicótica.

Esta pareja de guacamayos se encuentra ubicada en el centro de la isla, que tiene forma circular. Una vez el jugador se acerque a ellos, debe comenzar el juego. Este paso previo representa la llegada del jugador a la isla y sirve para habituarse a los controles del juego. Cuando el juego comienza, los guacamayos le explican brevemente al jugador su problema y en qué consiste el juego en una cinemática. Necesitan encontrar llaves para abrir su cofre del tesoro, pero están enterradas y no pueden recuperarlas. Para encontrar las llaves, los guacamayos dirán simultáneamente dos palabras que el jugador debe escuchar (escucha dicótica). En este instante se genera una serie de dibujos en la playa de la isla y dos de esos dibujos se corresponden con las palabras escuchadas, mientras que el resto no guardan relación. El jugador debe entonces buscar los dibujos que se corresponden con las palabras escuchadas y excavar en ellos para obtener las llaves. Este proceso se repite durante varias rondas ajustables y, al finalizar, los guacamayos logran abrir su cofre del tesoro si se han encontrado las llaves suficientes. Se decide que la ruta de trabajo a seguir establece que las primeras tareas que realizar son el desarrollo del terreno base y los algoritmos principales de las mecánicas de juego: movimiento, cámara y generación de dibujos aleatorios.

Este formato de juego, en el que el jugador primero escucha las palabras y después debe buscarlas en un grupo de dibujos que no ha visto previamente, se debe a que un ejercicio de escucha de estas características no debe tener ninguna ayuda visual previa, de manera que toda la comprensión resida exclusivamente en la escucha.

Para motivar al jugador, la recompensa por abrir el cofre es un objeto decorativo aleatorio para la isla, que se mantiene durante todo el proceso del juego. Por otro lado, se busca evitar que el jugador se frustre en caso de no acertar. Por esta razón, se añade un

margen de dos errores y la posibilidad de volver a escuchar las palabras en cualquier momento. Estas decisiones pueden ser de interés para los terapeutas, por lo que todos los errores y repeticiones realizadas se guardan en los resultados. Las palabras escuchadas y dibujos generados deben estar separados por número de sílabas para adecuarse a un ejercicio de escucha dicótica. Se decide, además, que estas palabras y dibujos sean aleatorios en cada sesión de juego. De esta forma, cada vez que se juegue se realiza un ejercicio totalmente nuevo.

Se tienen algunas consideraciones a la hora de elegir los dibujos que aparecen en la playa. En primer lugar, ninguna palabra debe guardar relación con la violencia o las armas, requisito por ser un juego para niños. Los dibujos generados deben ser suficientemente diferentes como para evitar confusiones debido a su similitud gráfica. Un ejemplo real de esto, ocurrido en el desarrollo del juego, son las palabras “hoja” y “pluma”. Su representación gráfica al utilizarse iconos simples y sin color resulta muy similar y puede haber confusiones. Además, las palabras y sus correspondientes dibujos deben ser sencillas, para garantizar que un niño las conozca y sepa identificarlas en un dibujo. Así, se escogen palabras básicas como nombres de animales, números, deportes, oficios y otros conceptos simples. La playa, zona donde se encuentran los dibujos, debe encontrarse totalmente vacía para que los dibujos puedan verse sin ningún impedimento. Se añaden objetos decorativos para ambientar en el centro de la isla, donde no se produce la acción del juego.

La necesidad de generar una escucha dicótica supone la limitación de que el juego debe jugarse siempre con auriculares, en caso contrario es muy difícil la diferenciación de las palabras escuchadas y no es una escucha dicótica. Una decisión derivada de esto consiste en que los ruidos distractorios presentes en el juego sean sonidos 3D (se aprecian mejor con auriculares), cuya percepción varía en función de la posición del jugador en la escena. De esta forma se le presenta al jugador una distracción mucho más realista, que debe tener en cuenta al moverse, pues la percepción de estos ruidos depende de ello.

Es necesario buscar una justificación para estos ruidos de distracciones. Se elige añadir un grupo de gaviotas, un barco, una radio y una fogata, pues son sonidos que guardan relación con una ambientación tropical o de playa. De igual forma, se añaden sonidos de ambiente que también guarden esta misma relación: olas, viento y música.

Se desarrollan pequeñas cinemáticas para mantener la atención del jugador en cada ronda y explicar el resultado obtenido.

En este proceso se requiere mucha atención del proyecto en conjunto, a través de reuniones con todos los integrantes del equipo. Esto se debe a la necesidad de que todos los módulos del juego tengan estructuras de niveles y rondas similares, con la finalidad de simplificar su entendimiento para el jugador. Esto puede suponer realizar cambios y ajustes imprevistos, pero destaca la importancia del trabajo en equipo. Cada integrante del grupo cuenta con una percepción del progreso del juego y del funcionamiento de los ajustes diferente, y es necesario unificar estas visiones en la medida de lo posible.

En resumen, la experiencia de juego final debe consistir en aparecer en una isla en la que se debe ayudar a dos guacamayos que dicen una palabra por cada oído. El jugador busca en la playa una serie de dibujos y debe excavar en aquellos que se corresponden con las palabras escuchadas.

5. Entorno de trabajo

Al tratarse de un proyecto multimedia, que combina aspectos de edición de audio, modelado 3D y diseño gráfico para la edición de imágenes, además de la propia programación del videojuego, se hace necesario utilizar diferentes softwares que abarquen todas estas disciplinas. En todo momento se establece la prioridad de utilizar software libre, para no aumentar los costes y que cualquier miembro actual o futuro del proyecto tenga acceso a seguir trabajando con los mismos materiales. Por esta misma razón también se hace necesario organizar y configurar el proyecto de manera que sea fácilmente entendible y manejable por otros compañeros en caso de ser necesario, así como clasificar todos los recursos utilizados de manera accesible.

5.1 Software utilizado

El entorno principal de trabajo es el motor de videojuegos Unity, donde se desarrolla todo el proyecto. Concretamente y para garantizar la compatibilidad de versiones entre todos los trabajos realizados por compañeros, se fija la versión LTS 2021 para utilizarse comúnmente. La versión de Unity utilizada es muy relevante, pues los proyectos realizados en versiones diferentes no son compatibles entre ellos. Unity además se apoya en Visual Studio 2019 para la programación de los *scripts* del videojuego en C#.

Por otro lado, para la creación del diferente contenido audiovisual presente en el videojuego se han utilizado los siguientes softwares:

- GIMP 2.10: programa de edición de imágenes digitales. Utilizado para la creación de elementos de la interfaz de usuario y edición de texturas.
- Blender 3.5: programa de modelado y animación. Utilizado para la creación y animación de modelos 3D, así como la modificación de materiales externos para adecuarlos a las necesidades del proyecto.
- Reaper 6.14: estación de trabajo de audio digital. Utilizado para la grabación y edición de los elementos sonoros presentes en el juego.

5.2 Configuración del proyecto

Para su correcta integración en el videojuego general de “El Planeta Sonoa”, cada uno de los módulos desarrollados debe tener una estructura similar que se divide en dos escenas de Unity, siendo una de ellas la escena principal de juego donde se desarrolla toda la acción y una escena auxiliar de menú, accesible únicamente si se juega en modo *offline* para poder configurar los parámetros de la partida como se haría a través de la web.

La carpeta “Assets” del proyecto también se encuentra organizada con el fin de localizar los diferentes recursos utilizados. Se crean subcarpetas para: animaciones, audio, materiales, modelos, *prefabs*, escenas, *scripts*, terreno, texturas e interfaz de usuario. En estas carpetas se irán añadiendo los recursos propios y externos correspondientes al módulo “Tropikus”.

6. Creación de terreno y ambientación

Dentro de la escena principal de juego se comienza por la creación de un terreno donde tiene lugar toda la acción. Debido a la ambientación de playa escogida, el terreno es muy sencillo y está formado únicamente por una pequeña isla circular. Toda la isla se rodea de un plano de agua, utilizando los *assets* del paquete externo Low Poly Water de la Unity Asset Store [17] (Figura 1). El jugador puede recorrer únicamente las zonas de tierra, habiendo un *collider* que impide el acceso más allá del agua.

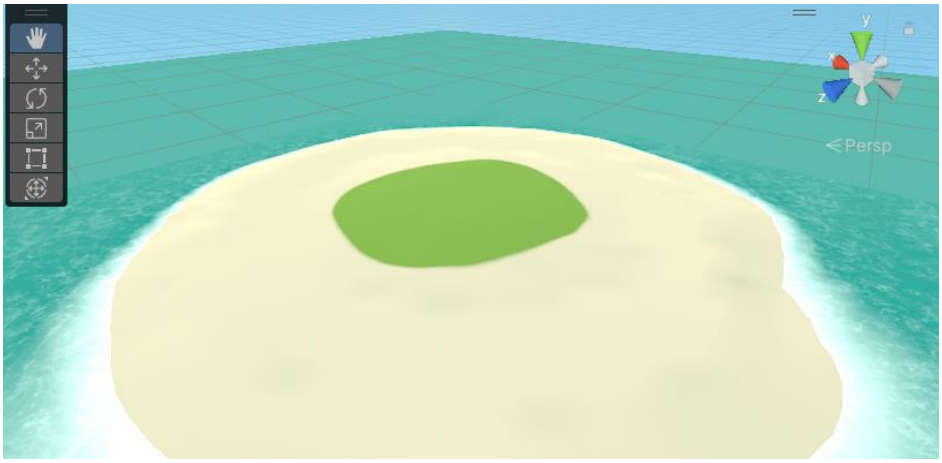


Figura 1: Isla creada.

Alrededor de la isla, recorriendo toda la playa, es la zona donde tienen que aparecer los dibujos que el jugador debe excavar. Para ello es necesario definir con antelación las posiciones en las que estos dibujos deben aparecer, creando un objeto invisible etiquetado como *spawn* (generación). Los dibujos se crean en forma de *sprites* (imágenes planas), por lo que se hace necesario aplanar el terreno en la zona donde deben aparecer. Para facilitar su visualización y colocación se ha añadido un *sprite* provisional con un número a cada *spawn* creado, que se utiliza únicamente para la edición del videojuego y no tiene ningún otro impacto (Figura 2).

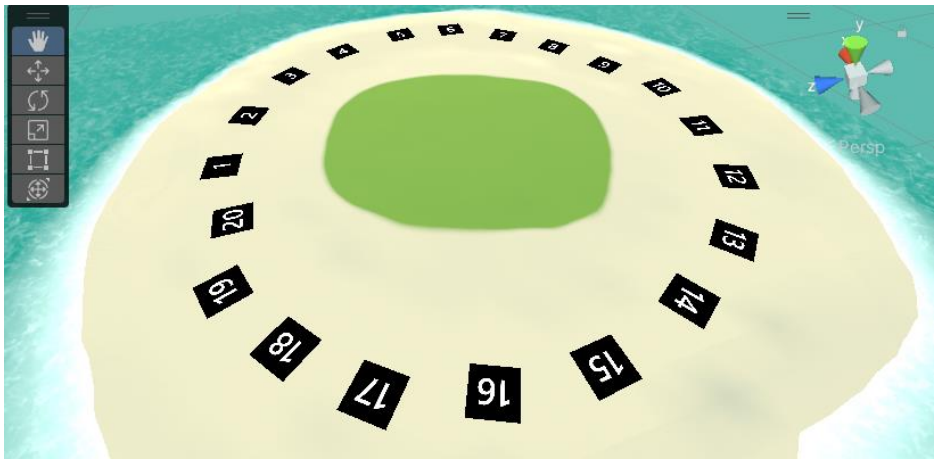


Figura 2: Colocación y numeración de *spawns*.

A continuación, se añaden objetos para decorar la isla obtenidos de diferentes packs de *assets* de la Unity Asset Store y que tienen únicamente una finalidad estética o de distracción. El objetivo es crear un pequeño asentamiento donde viven los guacamayos que guían al jugador (Figura 3).

Figura 3: Inserción de objetos a la isla.

Además de los objetos ya visibles, que se corresponden con el estado de la isla al inicio del juego, se añaden más decoraciones que no son visibles en un principio, pero se desbloquean en el progreso como recompensa al abrir el cofre de los guacamayos (activando el objeto correspondiente para hacerlo visible). La Figura 4 muestra la escena con estos objetos visibles.

Figura 4: Isla con objetos decorativos desbloqueables.

Con esto, se encuentra creado el mapa de juego principal a partir del cual se comienza a implementar el desarrollo técnico de *scripts* y otros objetos para su funcionamiento. Esta es la primera escena a la que accede el jugador al jugarse en modo *online*, sin embargo, es necesario crear la escena de menú a la que se accede en modo *offline* para poder configurar el juego sin acceder a la web.

7. Menú y ajustes

Con el objetivo de obtener la mayor versatilidad del juego, se ha buscado que este sea lo más ajustable posible. De esta forma el terapeuta que haga uso del juego puede adecuarlo a las necesidades de cada paciente, pudiendo definir niveles de dificultad personalizados y guardar diferentes sets de ajustes para su reutilización en otras ocasiones.

Los datos de los ajustes se almacenan en la clase `TROPI_GameSettings()`, pero pueden insertarse de formas diferentes según se juegue en modo *offline* u *online*. Se va a explicar utilizando como muestra el modo *offline*, ya que el ajuste a través de la web no es objeto directo de esta fase del proyecto (aunque su funcionamiento es idéntico). En el caso *offline* los ajustes se realizan mediante el menú auxiliar creado en la escena para tal efecto (Figura 5) y se almacenan en un objeto de ajustes persistente entre escenas mediante la sentencia `DontDestroyOnLoad(this.gameObject)` en el método `Awake()`.

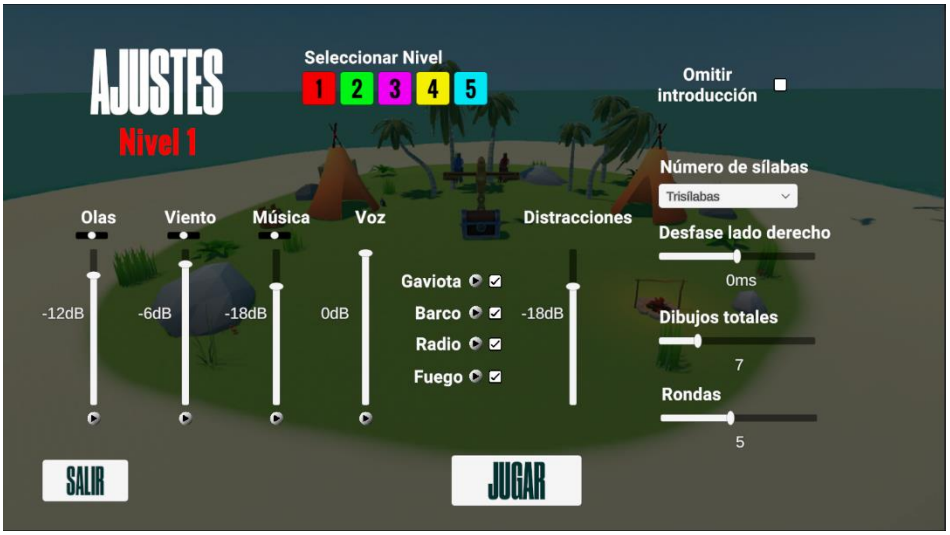


Figura 5: Menú de ajustes completo.

Para su organización, se han dividido los posibles ajustes en dos tipos: ajustes de sonido y ajustes de jugabilidad, aunque su funcionamiento, consulta y clases que los gestionan son comunes.

7.1 Ajustes de sonido

Existen diferentes sonidos en el juego, todos ellos con la finalidad de enmascarar el sonido de las palabras que deben ser escuchadas mediante ruidos para necesitar una mayor concentración y acercarlo más a una experiencia realista.

Los ajustes que pueden realizarse dependen del tipo de sonido, habiendo sonidos globales que se escuchan de igual manera en todo momento y sonidos 3D cuya percepción varía en función de la posición espacial del jugador. De los sonidos globales, que son cuatro (olas, viento, música y voz), puede ajustarse su nivel y, a excepción de la voz, su panoramización. De esta forma es posible que el jugador escuche ruidos concretos únicamente en un oído, algo que puede ser interesante para ciertos pacientes.

En el proceso del desarrollo surge la necesidad de modificar el funcionamiento de las fuentes de audio del juego, que se encuentran ahora separadas entre la escena del menú y del juego, debido a la necesidad de reducir la creación de objetos persistentes entre escenas. Durante el juego estos ajustes se gestionan y aplican mediante la clase `TROPI_SoundController()`, mientras que en el proceso de ajuste se gestionan mediante la clase `TROPI_MenuSounds()`. En ambos casos su funcionamiento se basa en la comprobación de la información de los ajustes realizados y su aplicación al canal del mezclador de sonido correspondiente a cada sonido (Figura 6).

Figura 6: Ajustes de sonidos globales.

En el caso de los sonidos 3D, llamados distracciones, únicamente es posible activarlos y desactivarlos y ajustar su nivel conjuntamente en un canal común del mezclador. Debido a su propia forma de funcionamiento no tiene sentido poder ajustar la panoramización en este caso. Se han implementado cuatro distracciones provenientes de diferentes objetos del juego: barco, gaviota, radio y fogata (Figura 7). El sonido en este caso lo emiten los propios objetos y se gestionan mediante la clase `TROPI_DistractVolume()` que comprueba en los ajustes si la distracción se encuentra activada o no.

Figura 7: Ajustes de distracciones.

El funcionamiento intrínseco al propio menú del modo *offline* (funcionamiento de sus botones, sonidos de prueba y valores mostrados) se gestiona mediante la clase `TROPI_MenuController()`, que solo se encuentra presente en esta escena.

Todos los ajustes de los niveles de sonido se aplican al mezclador que controla todas las salidas de sonido del juego, marcando el nivel máximo que puede alcanzar un canal en cada momento (Figura 8). En este caso, los ajustes únicamente permiten configurar el nivel de un canal entre -80dB y 0dB, si bien el mezclador es capaz de soportar una ganancia de hasta +20dB. Esta decisión, común a todos los módulos, responde a la posibilidad de que aparezcan distorsiones indeseadas en los sonidos al aplicar una ganancia superior a 0dB.

Figura 8: Mezclador del juego tras realizar ajustes.

7.2 Ajustes de dificultad

Estos ajustes se corresponden con el formato que tiene cada nivel de juego, eligiendo diferentes parámetros como el número de rondas y número de dibujos que aparecen en cada ronda. Estos parámetros son consultados a la hora de iniciar nuevos niveles. Se incluye también un selector para elegir el tipo de palabras que se escuchan en función de su número de sílabas, lo cual repercute en una mayor o menor dificultad, y el retardo entre lo escuchado en el oído izquierdo y derecho en milisegundos (Figura 9).

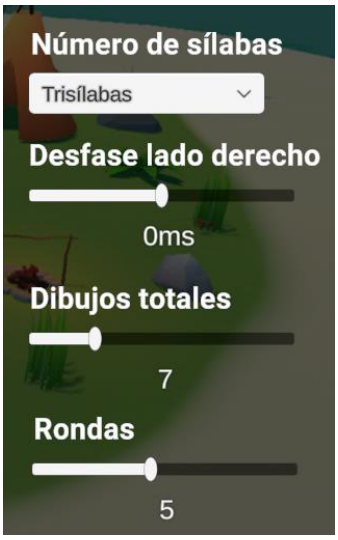


Figura 9: Ajustes de dificultad.

7.3 Selección de nivel

Pueden realizarse diferentes combinaciones de los ajustes vistos hasta el momento, añadiendo niveles de juego a la partida. Para ello se utiliza el selector correspondiente en el cual por defecto vienen desactivados todos los niveles distintos al 1, pudiéndose activar o desactivar libremente los niveles del 2 al 5 con otras configuraciones de ajustes. De esta forma, pueden realizarse ejercicios con dificultades distintas en una misma sesión de juego (Figura 10).

Figura 10: Selección y activación de niveles.

8. Desarrollo de mecánicas de juego

El funcionamiento básico del juego gira alrededor de un *script* central, de nombre `TROPI_PlayAgent.cs` (llamado en adelante `PlayAgent`), cuya función es establecer la comunicación entre todos los controladores y gestionar la secuencia de juego. Todas las clases consultan en `PlayAgent` información como el nivel actual o el estado de juego (pausa, viendo una cinemática, juego completado, esperando nueva ronda…) a través de variables públicas. De igual manera almacena información del progreso del jugador como el número de llaves recogidas, número de errores o la ronda de juego actual, entre otros. Por otro lado, `PlayAgent` se comunica con el resto de los controladores enviando mensajes cuando sea necesario realizar un nuevo proceso. A continuación, se detalla el funcionamiento de diversos aspectos y grupos de *scripts* con las diferentes clases utilizadas en "Tropikus", comunicadas a través de `PlayAgent`.

8.1 Player y cámara

Para gestionar el movimiento, interacción y cámara del jugador se han tomado como base *assets* del videojuego "Colorless" [18] realizado para la asignatura de Síntesis y Animación de Imágenes y se han adecuado a las necesidades de este proyecto. En este caso, y a diferencia de "Colorless", se trata de un juego en primera persona, por lo que el objeto de Unity que contiene al jugador será invisible y la cámara debe ubicarse a su altura para obtener una perspectiva de primera persona.

Se crean dos objetos: `Player`, que se instancia al iniciar el juego mediante `PlayAgent`, y `Camera` cuya misión es seguir al jugador para ver su punto de vista o enfocar a objetos en otros momentos a través de mensajes desde `PlayAgent`. Existen tres *scripts* principales que gestionan las acciones del jugador:

- `TROPI_MovPlayerDir.cs`: este *script* se encuentra insertado en el objeto `Player` y su función es gestionar, aplicando los ajustes elegidos, el movimiento del jugador (Figura 11) que se recibe al pulsar las diferentes teclas correspondientes al movimiento (a través del teclado o el mando) con `CharacterController` y `Move()`. Añade, además otras opciones de movimiento como saltar y correr. El salto genera un movimiento vertical ajustable a través de una variable de gravedad y correr aumenta la velocidad de movimiento al mantenerse pulsada su tecla en una cantidad también ajustable. Estas dos acciones de movimiento no son necesarias durante el juego, pero aportan una mayor flexibilidad al jugador para moverse por el terreno a su gusto.

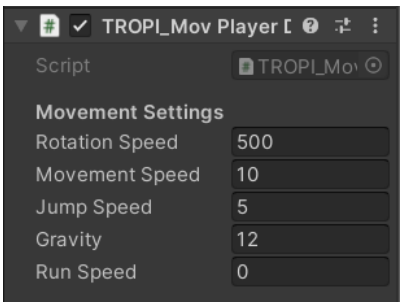


Figura 11: Parámetros de movimiento elegidos en el inspector.

- `TROPI_Camera.cs`: este *script* se encuentra incluido la cámara añadida en la escena y sigue al objeto que se le ordene con unos parámetros de altura y distancia ajustables. Recibe los movimientos de los ejes XY del ratón o mando para moverse alrededor de este objeto como se haya configurado y puede seguir o dejar de seguir nuevos objetos en mitad del funcionamiento del juego, de manera que a través de `PlayAgent` se llama a su función `FollowObject(GameObject obj)`, la cual toma como parámetro de entrada un objeto del juego, y recoloca la cámara siguiendo a este nuevo objeto obteniendo su posición y rotación.

- `TROPI_IntPlayer.cs`: este *script* se encarga de gestionar las interacciones del jugador con el entorno del juego cuando se pulsa una tecla asignada o se entra en un *collider*. Existen diferentes interacciones posibles en función del momento del juego, por ello, el *script* captura las pulsaciones en el teclado consultando a `PlayAgent` el estado de las variables correspondientes a cada situación del juego.
    - Se utiliza la tecla de interacción (tecla E en teclado o A en mando) para saltar la cinemática de introducción, si el jugador lo desea. En este caso y para evitar saltarla por error, la tecla debe mantenerse pulsada un segundo.
    - La misma tecla de interacción se utiliza para comenzar una nueva ronda cuando el juego lo indique.
    - La tecla de interacción también es la que debe pulsarse cuando el jugador decida excavar uno de los dibujos generados en la playa del juego.
    - La tecla de repetición (tecla R en teclado e Y en mando) se utiliza para volver a escuchar las palabras emitidas por los guacamayos, se encuentra únicamente activa en el transcurso de una ronda.
    - La tecla Escape en teclado o Menú en mando activa la pausa del juego.

  De forma pasiva, el jugador también interactúa con diferentes *colliders* de tipo *trigger*: uno de ellos es el encargado de iniciar la secuencia de juego cuando el jugador llega al centro de la isla y, por otro lado, cada uno de los dibujos de la playa contiene un *collider* de este tipo para comprobar que el jugador se encuentra lo suficientemente cerca como para excavarlo. Existe además un *collider* auxiliar para detectar si el jugador ha sobrepasado los límites del mapa de juego y recolocarlo, aunque nunca debería ser posible accederlo. Los *colliders* sólidos, además de impedir el movimiento del jugador, también son detectados para reproducir un pequeño efecto de sonido de choque (Figura 12).

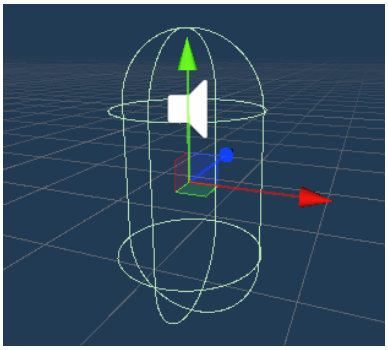


Figura 12: Elementos del objeto `player`.

Como resultado de estos tres objetos se controla la entrada de todas las interacciones del jugador con el juego. Se resumen de forma conjunta en la Tabla 2.

A raíz de las reuniones mantenidas por todo el equipo, se realizan ajustes en el movimiento del jugador para hacerlo más ágil, ajustando la velocidad de la cámara y la sensibilidad de movimiento. Esta es la razón por la que se decide añadir la acción de correr, para aquellos jugadores que deseen moverse con más rapidez. En este punto se aprecia la importancia de realizar pruebas del proyecto en equipos diferentes para ver el funcionamiento del juego con cambios en las configuraciones. Por ejemplo, desarrollar el juego con una sensibilidad de ratón muy alta supone que aquellas pruebas realizadas en un ordenador con una menor sensibilidad tienen un movimiento excesivamente lento.

Tabla 2: Asignación de botones.

Acción	TECLA	
	Teclado y ratón	Mando
Movimiento	W A S D	Joystick izquierdo
Movimiento de cámara	Movimiento de ratón	Joystick derecho
Interacción	E	A
Repetición de palabras	R	Y
Salto	Barra espaciadora	LB
Correr	Mayús. izquierda	RB
Pausa	Escape	Menú

8.2 Generación de dibujos

La mecánica principal del juego se centra en la generación de dibujos aleatorios secuencialmente alrededor de la isla. Estos dibujos, que se regeneran en la playa en cada ronda de juego, están constituidos a través de un *prefab*. Un *prefab* es un conjunto de objetos previamente agrupados en un solo paquete que actúa como una plantilla a partir de la cual se pueden crear nuevas instancias del objeto en la escena. Así, cada vez que se crea un nuevo dibujo contiene todos sus elementos asociados. Actualmente, el juego cuenta con una batería de 75 *prefabs* de este tipo, habiendo 25 para cada número de sílabas.

Dichos *prefabs* incluyen el *sprite* correspondiente al dibujo que representa la palabra escuchada, un *sprite* que representa que ya ha sido excavado y un *sprite* con un indicador para informar al jugador de que es posible excavar el dibujo pulsando la tecla de interacción que se activa al entrar en su rango y accionar su *collider* (Figura 13). Incluye también el audio con el sonido de la palabra correspondiente al dibujo con voz sintética, cuyo proceso de generación se detalla en el Anexo II.

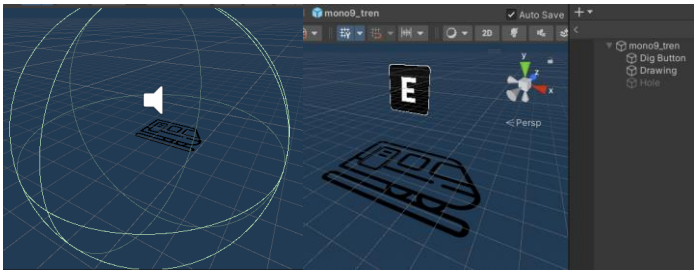


Figura 13: Ejemplo de un *prefab* creado. Izquierda: botón de interacción, derecha: *collider* y fuente de audio.

Además de estos *sprites,* cada *prefab* de los dibujos incluye su fichero de audio con la palabra grabada y un *script* de nombre `TROPI_CorrectController.cs` que maneja los diferentes componentes del *prefab.* Esta clase incluye un identificador con el nombre del dibujo en una variable pública tipo *string* y un booleano público que indica si el dibujo correspondiente es correcto o no, es decir, si es una de las palabras que el jugador ha escuchado o no (Figura 14).

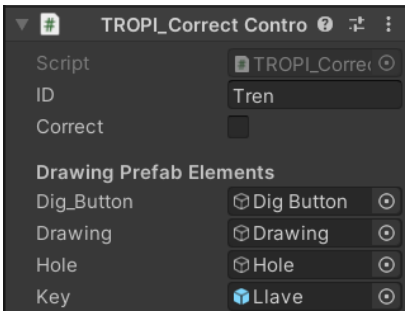


Figura 14: Elementos de un *prefab* de dibujo en el inspector.

Todos estos *prefabs* de dibujos se instancian mediante otro *script* principal que los gestiona todos, llamado `TROPI_GenerateDrawings.cs`. El funcionamiento de este *script* es uno de los más complejos del juego y necesita especial atención, ya que todo el sistema de juego se basa en él.

En primer lugar, esta clase incluye todos los objetos de la escena correspondientes con los *spawn* y los almacena en un *array*. Estos serán los posibles lugares donde pueden aparecer dibujos en la arena. Por otro lado, almacena como variables todos los *prefabs* correspondientes a cada uno de los dibujos separados según su número de sílabas. Cada vez que se llama a su función `NewLevel(int syllable_number)` carga en un *array*, en función del número de sílabas escogido como parámetro de entrada, todos los *prefabs*.

Una vez se llama a su función principal, `GenerateNumbers()`, la clase crea una lista de números de nombre `numbers` del tamaño total del *array* de *prefabs*. Genera un número aleatorio `randomStart`, este número será el punto de inicio con el que se recorre el *array* de *spawns*, haciendo así aparecer los dibujos en lugares diferentes de la isla en cada ronda de juego, pero siempre juntos. A continuación, se incluye un bucle `for` que va generando números aleatorios entre el 0 y el tamaño de la lista `numbers`, instanciando el *prefab* correspondiente a ese índice en el *array*. De esta forma se genera la cadena de dibujos visibles en la playa que el jugador debe excavar (Figura 15). En la escena pueden verse identificados como “clones”, haciendo referencia a que son instancias basadas en un *prefab* (Figura 16).

Figura 15: Vista de dibujos generados en el suelo.

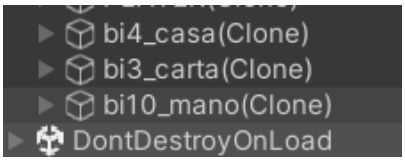


Figura 16: Vista de *prefabs* instanciados aleatoriamente en la escena.

Se elimina el índice del *prefab* instanciado de la lista creada. De esta forma se evitan las repeticiones y cada dibujo se instancia una única vez, ya que el número correspondiente a su índice ya ha sido eliminado de la lista en la siguiente interacción del bucle. Esto se repite tantas veces como número de dibujos se haya seleccionado en los ajustes. Como resultado se genera una cadena de dibujos únicos en un lugar aleatorio de la isla (Figura 17 y Figura 18).

Figura 17: Ejemplo de generación de una cadena de tres dibujos con comenzando en la posición 17.

Figura 18: Ejemplo de generación de una cadena de tres dibujos con comenzando en la posición 4.

El último paso es elegir dos de los dibujos que se hayan generado como los dibujos correctos. Para ello se toman todos los dibujos presentes en el juego, se almacenan en un *array* y se seleccionan dos aleatoriamente. Se llama a la función `SetAsCorrect()` de cada uno de los dibujos que activa el booleano reservado para indicar que el dibujo es correcto y se reproduce el fichero de audio incluido en estos (Figura 19).

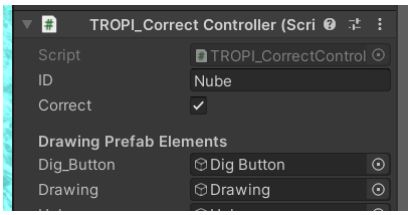


Figura 19: Variable pública `correct` activada en un dibujo elegido como correcto.

En el momento en el que el jugador ha escuchado las dos palabras correspondientes a los dibujos correctos, puede comenzar a jugar buscando los dibujos en la playa. Para excavar un dibujo, como se ha mencionado anteriormente, el jugador debe acercarse a una distancia suficiente y pulsar la tecla de interacción. En ese momento se llama a la función `Dig()` del *script* incluido en el dibujo, que comprueba si el dibujo excavado fue marcado como correcto o no, aumentando el contador de dibujos excavados correctos o incorrectos de `PlayAgent` respectivamente, mediante los cuales gestiona el progreso de la ronda. Se resume, en forma de diagrama de flujo, el funcionamiento de la lógica tras la generación de dibujos (Figura 20).

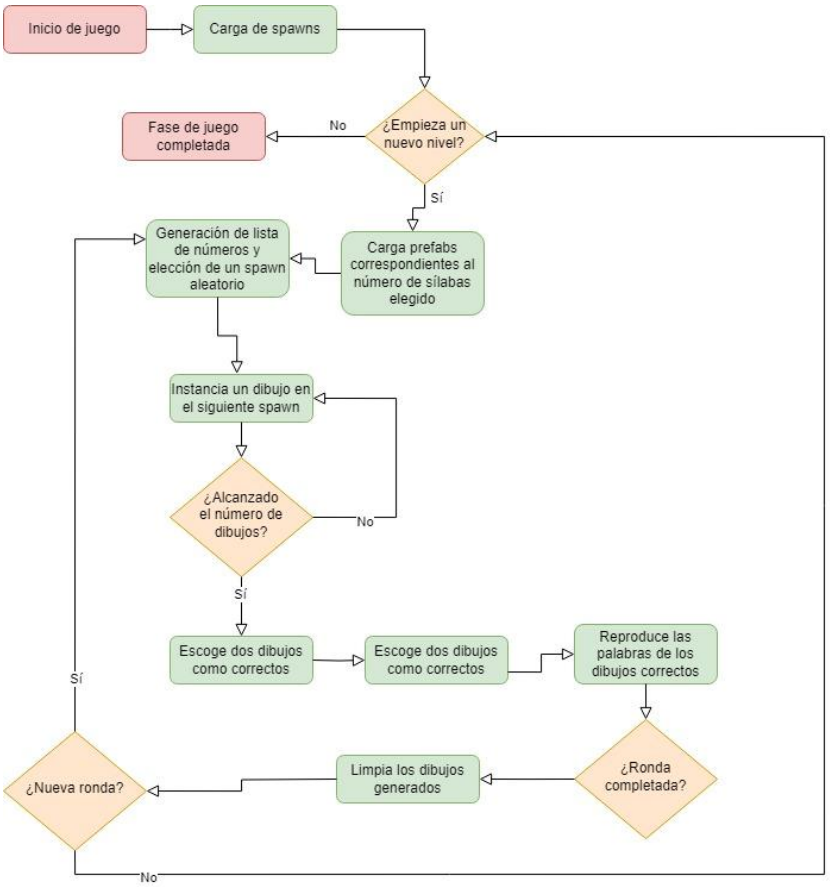


Figura 20: Diagrama de flujo de la generación de dibujos.

9. Integración de modelos animados

En el videojuego aparecen diferentes objetos animados para dar vida y realismo a la experiencia. En el caso de este módulo todos los modelos 3D animados utilizados son importados de diferentes creadores y repositorios *online*, si bien ha sido necesario realizar modificaciones para adecuarlos a las necesidades del juego. Se van a detallar los diferentes modelos utilizados en el juego que contienen animaciones, existen otros objetos que se mueven en la escena, pero no son realmente objetos animados y únicamente contienen *scripts* que varían su posición o rotación.

9.1 Gaviotas

Uno de los componentes animados del juego es un grupo de gaviotas que revolotean en lo alto de la isla. Su finalidad es ser una de las distracciones del jugador, emitiendo ruidos desde arriba. El objeto que contiene las gaviotas se mueve en distintas direcciones para variar la fuente de sonido independientemente de la animación del modelo (Figura 21), que únicamente aporta estéticamente.

Figura 21: Modelo de gaviotas.

El objeto del modelo de las gaviotas, que se ha obtenido de Sketchfab [19] está compuesto por cuatro de ellas (Figura 21), todas idénticas en distintas posiciones y cuya animación comienza con el inicio del juego y se mantiene en bucle constantemente durante todo el tiempo, como así lo refleja su diagrama del controlador de animaciones de Unity (Figura 22).

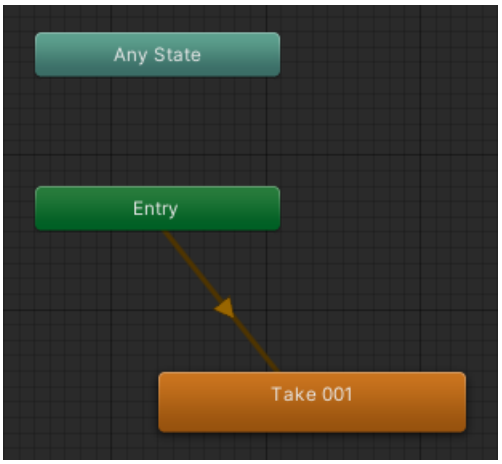


Figura 22: Diagrama del controlador de animación de las gaviotas.

9.2 Cofre

El motivador principal del jugador para jugar al juego es buscar llaves que le permitan abrir un cofre y obtener recompensas. Para reflejar este hecho y generar una experiencia de juego más satisfactoria, es necesario que el cofre se abra realmente, y para ello se utilizan animaciones en el modelo, obtenido en Turbosquid [20]. La animación es muy sencilla y únicamente consiste en el movimiento de una de las partes del modelo del cofre que se corresponde con la tapa para dar una sensación de apertura (Figura 23).

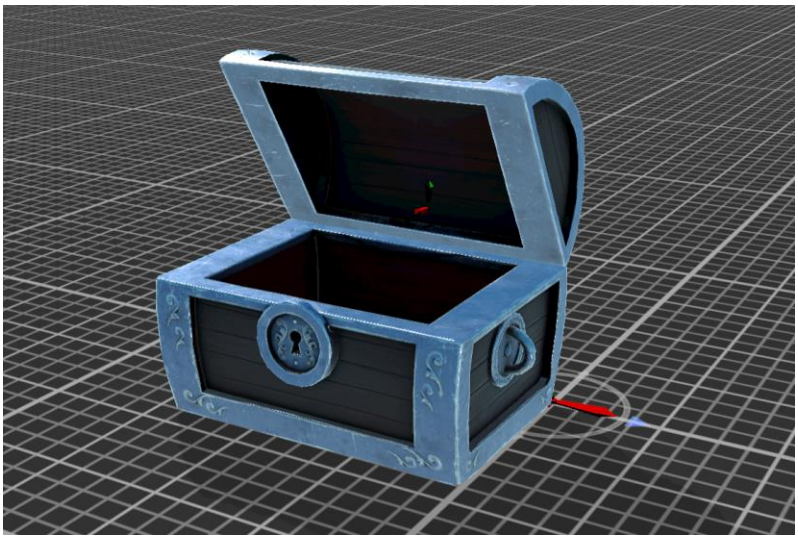


Figura 23: Cofre en animación de apertura.

Esta animación se inicia cada vez que el jugador finaliza un nivel habiendo obtenido las llaves necesarias y desemboca en un fundido a blanco para enfocar el objeto que se ha desbloqueado en esa ocasión. De igual manera, el cofre se cierra para iniciar un nuevo nivel. Para esto el controlador de animación de Unity cuenta con tres estados (Figura 24). El estado “closed” no realiza ninguna animación y únicamente cuenta con una transición al estado “open”, en el que el cofre se abre y se queda en esa posición. Por último, se realiza una transición inversa a la anterior para cerrarse mediante el estado “close”.

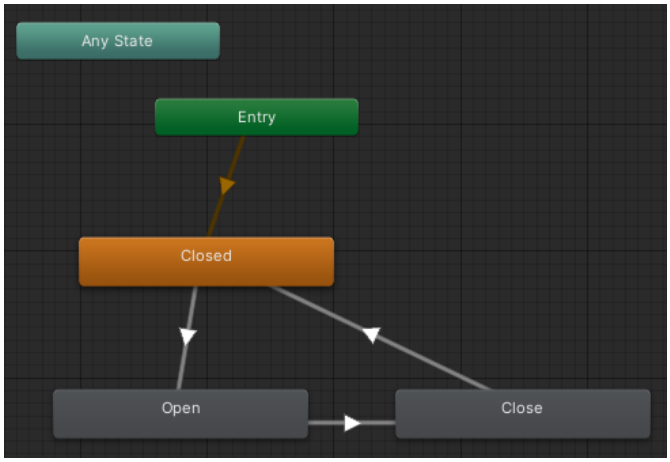


Figura 24: Diagrama del controlador de animación del cofre.

El modelo se coloca en el centro de la isla, al pie de un tótem en el que se apoyan los guacamayos, personajes principales del juego (Figura 25). En el proceso de juego, las animaciones del cofre se activan mediante su clase asociada `TROPI_AnimateChest()`, que recibe mensajes desde `PlayAgent` para activar las distintas animaciones.

Figura 25: Colocación del cofre en la escena.

9.3 Guacamayos

La pareja de guacamayos son los únicos NPC (*Non-Playable Character*) presentes en el juego juego y, por tanto, sus animaciones son más complejas e importantes para el jugador. Esto se traduce también en una mayor necesidad de adaptar los modelos originales.

El modelo, creado por Alien Interactive para Turbosquid [21], disponía únicamente de una animación integrada, por lo que ha sido necesario editarlo para crear diferentes animaciones nuevas. Estas animaciones se crean editando el modelo .fbx del guacamayo en Blender, añadiéndolas en el editor de acciones y grabando los nuevos movimientos en la línea de tiempos (Figura 26).

Figura 26: Creación de acciones de animación del modelo del guacamayo en Blender.

Así, el modelo cuenta ahora con cuatro acciones diferenciadas (Figura 27):

- Animación de espera (*idle*): por defecto cuando no ocurre ninguna acción:
- Animación de habla: durante la introducción, mientras los guacamayos explican su problema.
- Animación de negación: cuando el jugador excava dibujos incorrectos.
- Animación de vuelo: celebración, cuando se excavan dibujos correctos y cuando finaliza el juego.

El modelo conteniendo estas acciones se exporta en un nuevo fichero .fbx.

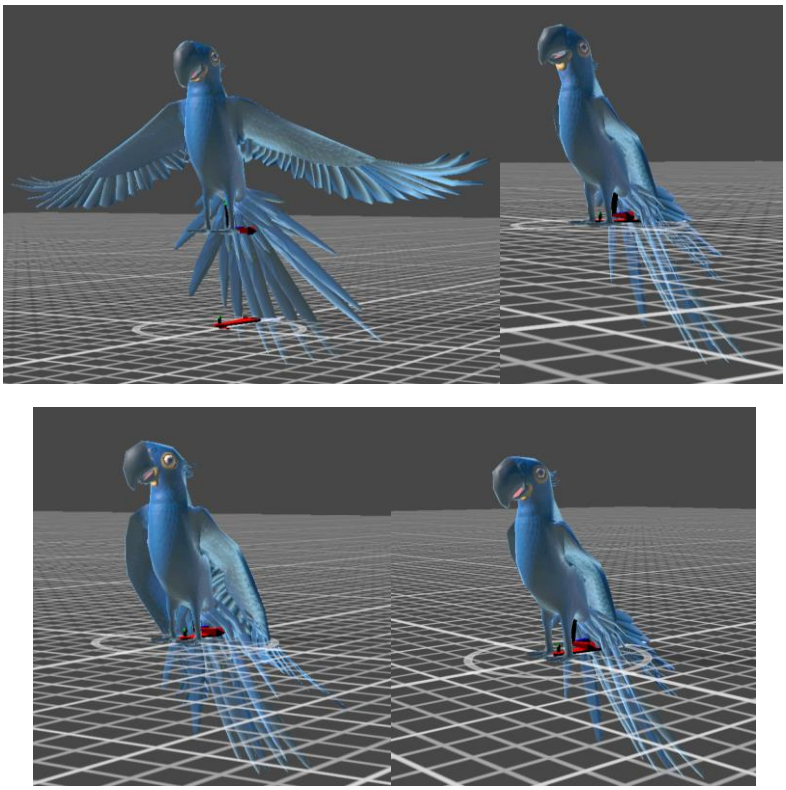


Figura 27: Estados de animaciones creados. Superior izquierda: vuelo, superior derecha: hablar, inferior izquierda: negación, inferior derecha: espera.

En el diagrama del controlador de animación de Unity (Figura 28) asociado a los guacamayos, se observa que al inicio se activa la animación *idle* y se mantiene en ella hasta ser activada cualquier otra en el momento en el que el jugador inicia una nueva acción. Por esta razón, a todas las demás animaciones se debe poder acceder desde el estado idle y contar con una transición preparada para ello. Estas transiciones deben ser rápidas para que la animación se ejecute de forma instantánea al momento en el que se le solicita. Además, la animación de habla también está preparada para pasar directamente a las de negación y vuelo. La configuración de estas transiciones se realiza en previsión de posibles interrupciones de la animación de habla.

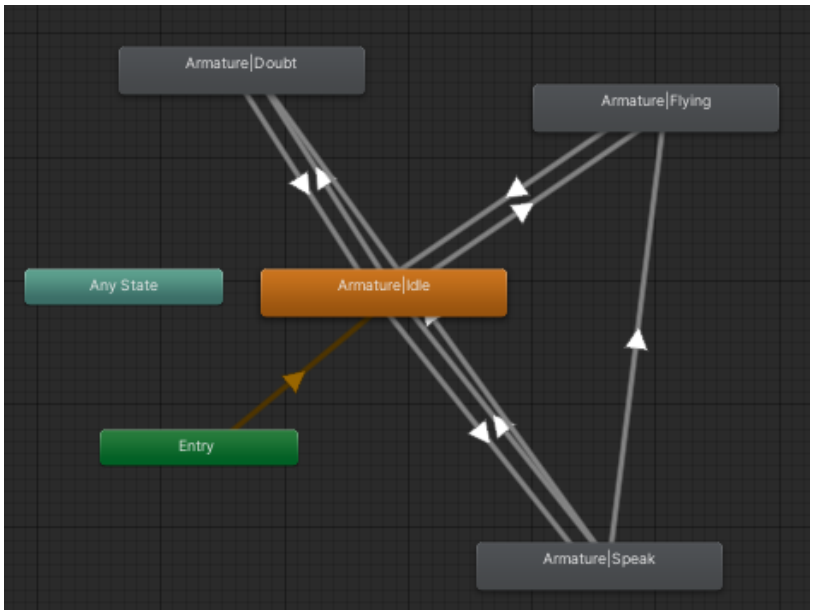


Figura 28: Diagrama del controlador de animación de los guacamayos

Las animaciones no han sido el único aspecto de estos modelos que se ha modificado para este juego. El modelo original se correspondía con un guacamayo de color azul, pero era necesario incluir una pareja de guacamayos (ya que el objetivo es representar que cada uno te habla en un oído) y utilizar dos modelos idénticos puede resultar poco llamativo. Por esta razón, se ha creado un segundo modelo de guacamayo clonando el original e importando nuevas texturas editadas para ser de color rojo.

Este proceso se realiza haciendo uso en primer lugar del software de edición de imágenes Gimp. En él se editan los ficheros que contienen las texturas del modelo inicial del guacamayo azul y se colorean al rojo mediante la función de coloreado disponible en este programa. Sin embargo, el resultado obtenido únicamente con este paso no es deseable debido a que todas las partes del modelo del guacamayo cambian su tono cromático a tonos rojizos y solamente se desea cambiar su plumaje. Para esto, se toman de nuevo las texturas originales y se colocan en una capa inferior. A continuación, se borran de las texturas las partes cuyo color no debe verse alterado (ojos, pico, patas…) para dejar ver de nuevo sus colores originales (Figura 29).

Figura 29: Comparativa de la textura original (izquierda) y la editada a rojo (derecha).

Una vez se han generado los archivos con las nuevas texturas, se edita en Blender una copia del modelo 3D del guacamayo (Figura 30). En el apartado de texturas se importan los ficheros previamente creados para aplicarlos al modelo. El resto de los parámetros de la textura se dejan intactos para que su comportamiento se idéntico al del modelo azul. El modelo se exporta en un nuevo .fbx independiente del guacamayo azul, aunque en Unity comparten controlador de animación.

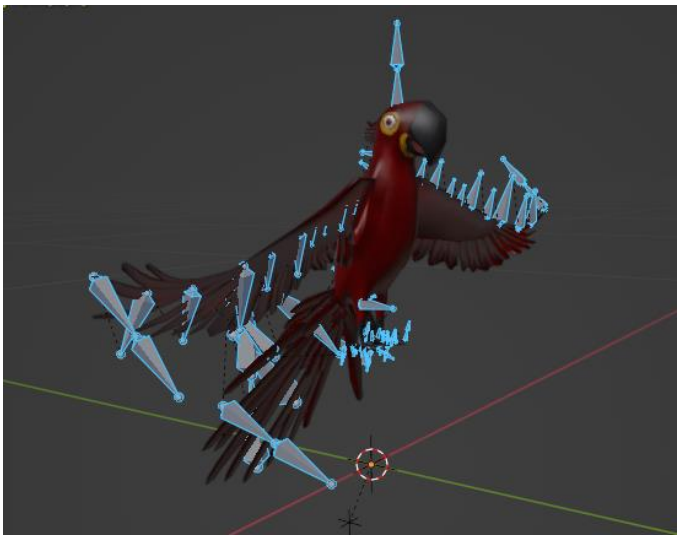


Figura 30: Nueva textura roja importada en el modelo en Blender.

Como resultado se obtiene una pareja de guacamayos, azul y rojo, que se encuentran colocados en el centro de la isla (Figura 31) y que son los encargados de aportar información al jugador con sus animaciones y diálogos. Para ello, se controlan los estados de animación mediante el *script* `TROPI_AnimateParrot.cs`, cuya finalidad es activar y desactivar las diferentes variables que controlan los estados mediante mensajes recibidos desde `PlayAgent`. Además, los guacamayos cuentan con una fuente de audio para reproducir las líneas de diálogo que también se activan desde esta clase. Estos diálogos han sido grabados con doblaje real, proceso que se detalla en el Anexo II.

Figura 31: Colocación de los guacamayos en la escena.

10. Secuencia de juego

El juego se divide en rondas y niveles como requisito de diseño. Cada acción de escucha de dos palabras y búsqueda de sus dibujos correspondientes conforma una ronda de juego. Un nivel es una sucesión de rondas con los mismos ajustes. En la configuración del juego es posible activar entre uno y cinco niveles para una misma sesión de juego, aplicando ajustes distintos en cada uno de ellos, y cada nivel puede contener entre una y diez rondas. El juego comienza con una cinemática introductoria que da paso a la sucesión de rondas y niveles configurados y que, una vez completada, activa una nueva cinemática de cierre para finalizar el juego. A continuación, se resume el funcionamiento de la secuencia de juego a modo de diagrama de flujo (Figura 32).

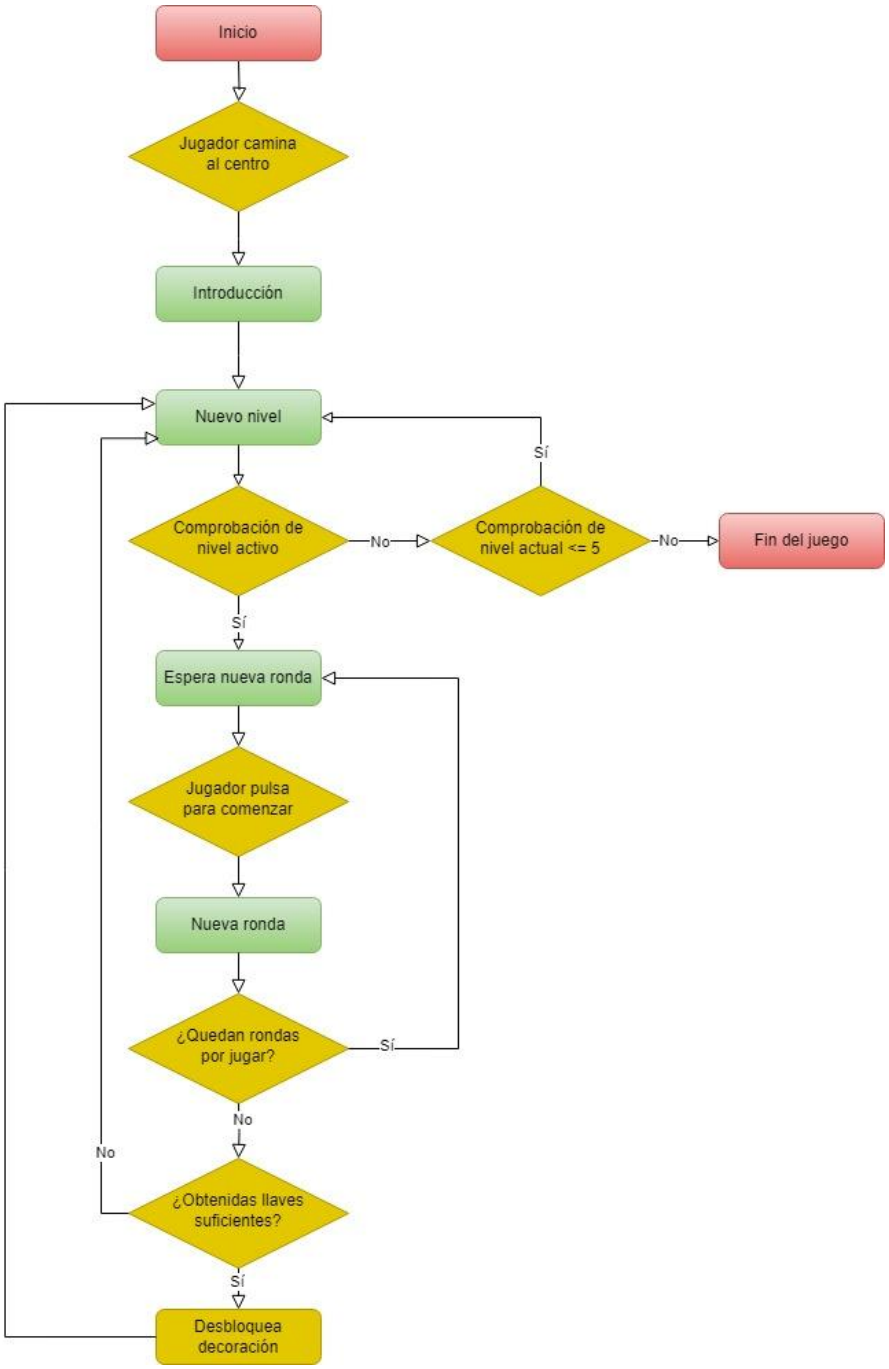


Figura 32: Diagrama de flujo de la secuencia de juego.

La mayor parte de la gestión de la secuencia de juego, que controla el progreso entre rondas y niveles del juego en función de los ajustes seleccionados, se realiza en la clase principal `PlayAgent`.

`PlayAgent` contiene una variable denominada `current_level` que indica en qué nivel de juego se encuentra en ese momento. Al comenzar la acción de juego se llama a la función `NextLevel()` de `PlayAgent`, que comprueba cuál es el siguiente nivel al que debe entrar al juego mediante la variable `current_level`. Una vez se ha iniciado el nivel correspondiente comienza la primera ronda del nivel.

El funcionamiento de la gestión de rondas está dividido en varias funciones, en orden de aparición son:

1. `WaitForRound()`: al comienzo de cada ronda se recoloca al jugador en el centro de la isla, bloquea su movimiento y se espera a que pulse una tecla para escuchar las dos palabras de la ronda. Esta espera fue un requisito a petición de las terapeutas tras una reunión, con el fin de poder garantizar que el paciente escucha con atención las nuevas palabras de la ronda.
2. `RoundReady()`: función que se activa cuando el jugador pulsa la tecla para comenzar. Devuelve el movimiento al jugador y da la orden de comenzar la nueva ronda.
3. `NewRound()`: se inician todos los contadores que controlan la ronda y se llama la función para generar los dibujos de la clase `GenerateDrawings()` ya detallada.
4. `UpdateKeys()`: se llama cada vez que el jugador encuentra una nueva llave, es decir, excava un dibujo correcto. Incrementa el contador de llaves y envía la orden de actualizar la interfaz.
5. `RoundCompleted()`: una vez se han encontrado las dos llaves o se ha alcanzado el número máximo de errores, la ronda está completa. Se activa una animación según se hayan conseguido las llaves o no y se comprueba si la ronda completada es la última del nivel para dar por finalizado el nivel o comenzar una nueva ronda (repitiendo todos los pasos anteriores).

En el caso de haberse completado el número de rondas, finaliza el nivel activando la función `LevelCompleted()`. De igual forma se activa una animación en función del resultado del nivel, en el caso de haberse completado con éxito se llama mediante la función `UnlockDecoration()` a la clase auxiliar `TROPI_DecorationController()` que desbloquea al azar un objeto decorativo para la isla y activa la animación de apertura del cofre.

Comprobando de nuevo mediante la función `NextLevel()`, se da comienzo al siguiente nivel o finaliza el juego si ya era el último. En este punto se guardan todos los resultados de la partida obtenidos por el jugador para su posible análisis posterior. Paralelamente también es posible interrumpir el progreso del juego y salir de este mediante el menú de pausa, que se activa cuando el jugador pulsa la tecla correspondiente en cualquier momento del juego. Si el juego se interrumpe de esta manera también se guardan los datos obtenidos hasta ese momento.

11. Clases auxiliares

11.1 Interfaz de usuario

Existen otros *scripts* para controlar diferentes funcionamientos del juego, siendo el principal de ellos `TROPI_CanvasController.cs`. Esta clase gestiona todos los diferentes *canvas* que aparecen en el juego, añadidos mediante el inspector. Esto incluye el HUD (Heads-Up Display), mensajes para guiar al jugador, el menú de pausa, el aviso de final de juego y un *canvas* para transiciones. Gestiona su activación y desactivación, así como la información mostrada en texto en el caso del HUD.

El HUD, consta de dos apartados en las esquinas superiores izquierda y derecha. A la izquierda, se muestra un contador con las llaves recogidas en el nivel, de lo cual depende la obtención de nuevas decoraciones para la isla. A la derecha, se muestra un indicador que informa al jugador del número de rondas que componen el nivel actual y de la ronda jugándose en ese instante (Figura 33). La clase `TROPI_CanvasController()` consulta en los ajustes y en `PlayAgent` la información del estado de juego para actualizar estos indicadores.

Figura 33: HUD.

Existen diversas guías en forma de texto que aparecen en el progreso del juego para aportar indicaciones al jugador de las acciones que debe o puede realizar. Únicamente se activan o desactivan en función del momento del juego. Existe una guía para informar al jugador de que debe pulsar una tecla para comenzar la ronda (Figura 34), una para omitir la introducción cuando está en curso (Figura 35) y otra para informar de que puede pulsar una tecla para escuchar de nuevo las palabras (Figura 36).

Figura 34: Guía de indicación de inicio de ronda.

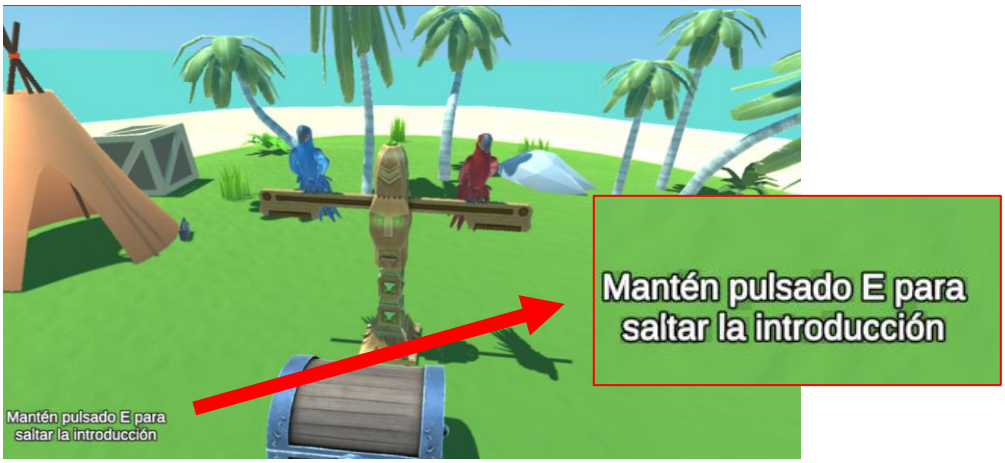


Figura 35: Información para saltar introducción.

Figura 36: Información para volver a escuchar las palabras.

El menú de pausa (Figura 37) puede activarse en cualquier momento del juego pulsando la tecla correspondiente para ello. En este momento, se para la escala de tiempo del juego y se pausa la reproducción de todos los audios de diálogo, mientras se activa un *canvas* que muestra el menú de pausa. Incluye un botón para continuar el juego y otro para abandonarlo, cargando de nuevo la escena del menú general en "El Planeta Sonoa".

Figura 37: Menú de pausa.

Con un funcionamiento similar a la pantalla de pausa, aparece una pantalla al final del juego para ofrecer volver al menú principal (Figura 38).

Figura 38: Pantalla de final de juego.

El juego es compatible para jugar con teclado y ratón o con mando de videoconsola, esto significa que las teclas que debe pulsar el jugador pueden variar en función del dispositivo que está utilizando y, por tanto, las guías que muestra el juego indicando la tecla que debe ser pulsada también deben actualizarse. Para esto se utiliza la clase `InputDetector()` que se encuentra integrada en el proyecto general. Su función es comprobar si hay un mando conectado para activar una variable pública de tipo booleano que se puede consultar y utilizar para crear condiciones con el objetivo de activar diferentes *canvas* o *sprites* según su valor (Figura 39). Como se trata de una clase incluida en el proyecto global del juego, esta es una funcionalidad que no se encuentra disponible en ejecutables independientes del módulo "Tropikus".

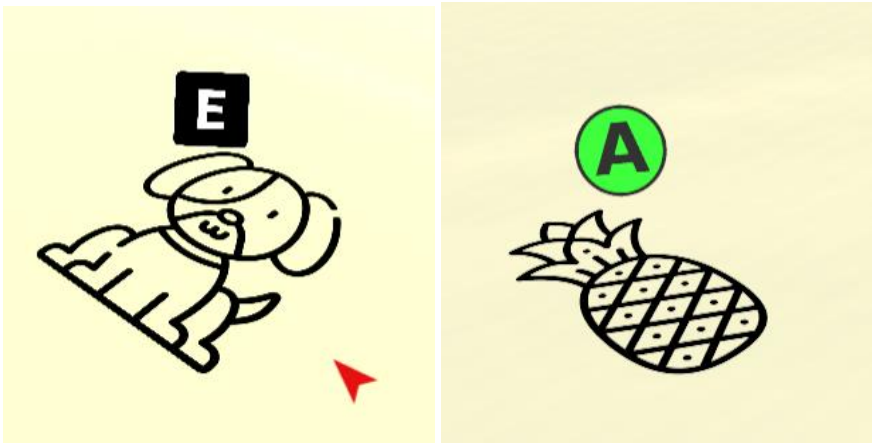


Figura 39: *Sprite* de botón de interacción en juego con teclado (izquierda) y mando (derecha).

11.2 Distracciones

Los objetos encargados de producir ruidos a modo de distracciones incluyen una clase auxiliar para comprobar en los ajustes del nivel actual si el ruido correspondiente al objeto está activado o no, esta clase es `TROPI_DistractVolume()`.

Como se ha visto en la pantalla de ajustes (Figura 5) existen cuatro objetos de estas características y todos ellos comparten el mismo funcionamiento. Para identificar el objeto se ha añadido un selector mediante el cual puede asignarse el tipo de objeto para que compruebe su variable correspondiente en los ajustes (Figura 40).

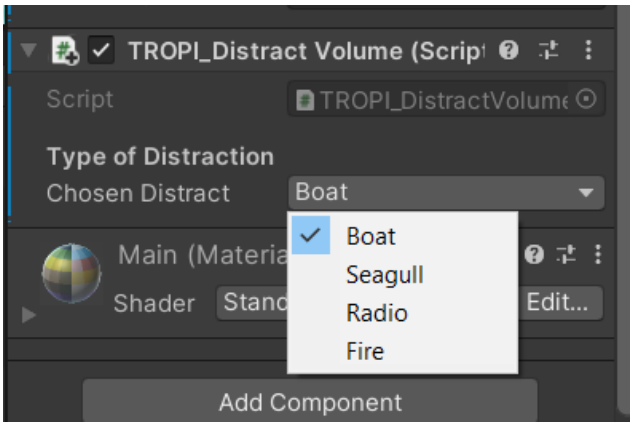


Figura 40: Selección del tipo de distracción en el inspector.

De esta forma solo es necesario implementar una clase independientemente del tipo de objeto. En caso de estar activo en el nivel, se le asigna un volumen distinto de 0 a la fuente de audio del objeto, que está configurada para ser una fuente de audio espacial y, por tanto, debe ir incluida en el propio objeto. De esta forma el paciente percibirá los ruidos en función de su posición en la isla. Concretamente el barco emite el ruido de bocina en movimiento alrededor de la isla, las gaviotas emiten un ruido desde el cielo y la radio y la fogata emiten ruidos similares al ruido blanco desde puntos en el suelo de la isla (Figura 41).

Figura 41: Colocación de objetos de distracción. Superior izquierda: barco, superior derecha: fogata, inferior izquierda: gaviotas, inferior derecha: radio.

11.3 Movimiento de objetos

Otros *scripts* auxiliares tienen la finalidad de realizar movimientos de objetos en la escena para aportar dinamismo al juego. Estos son:

- `TROPI_LinearMovement.cs`: genera un movimiento lineal cíclico de un objeto en unas coordenadas relativas dadas y con una velocidad especificada. Se utiliza en las gaviotas que vuelan por encima de la isla y en todos los objetos que se encuentran en el mar para simular el efecto del oleaje (Figura 42).

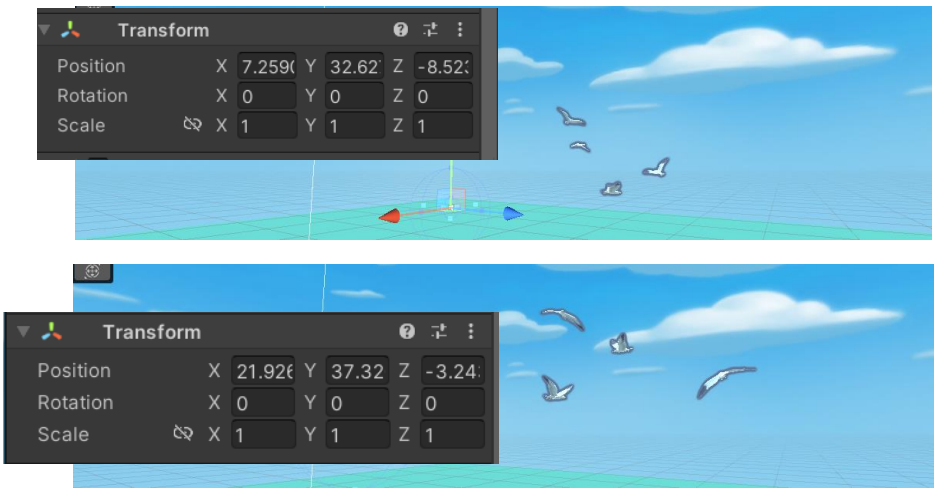


Figura 42: Comparativa de movimiento en dos instantes del objeto de las gaviotas.

- `TROPI_CircleMovement.cs`: genera el movimiento circular de un objeto alrededor de otro objeto añadido en el inspector. Se utiliza en el barco que aparece de fondo, que gira alrededor de la isla y cuyo sonido es una de las distracciones configurables (Figura 43).

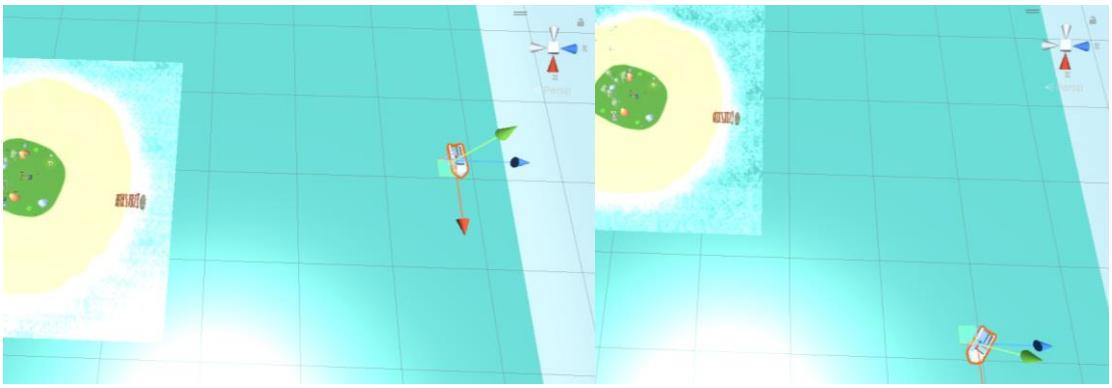


Figura 43: Comparativa del movimiento en dos instantes del objeto del barco.

- `TROPI_LookAtCamera.cs`: rota un objeto para que se mantenga siempre mirando a la cámara. Se utiliza en el *sprite* que aparece encima de cada dibujo para indicar al jugador la tecla que debe pulsar para excavar, cuando este se encuentra dentro de su rango, para que siempre sea legible (Figura 44).

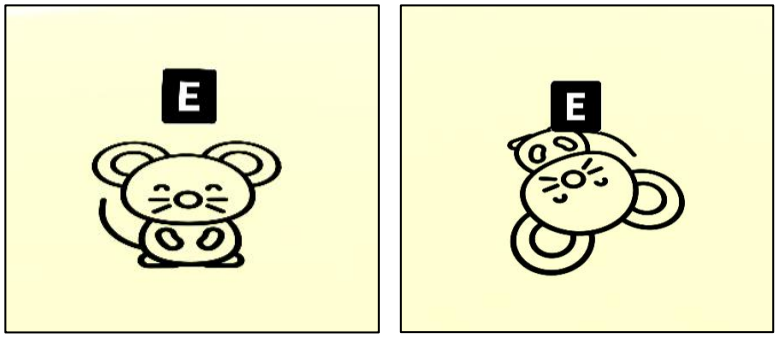


Figura 44: Movimiento del *sprite* de la tecla de interacción en función del movimiento de la cámara.

- `TROPI_CanvasFadeIn.cs`: genera un *fade in* y un *fade out* en el *canvas* al que está asignada la clase. Se utiliza en la transición entre la animación de apertura de cofre y el momento que muestra el objeto desbloqueado (Figura 45).

Figura 45: Transición a blanco.

12. Integración en el proyecto global

Alcanzado este punto del proyecto, el módulo de "Tropikus" se encuentra desarrollado por completo y es funcional de manera independiente. El siguiente paso es su integración en el juego base "El Planeta Sonoa" como uno de los juegos seleccionables y configurable a través de su página web. Esto puede ser causa de incompatibilidades o redundancias no contempladas previamente.

Por ejemplo, la razón por la que todos los *scripts* que forman parte de "Tropikus" comienzan con el prefijo "TROPI" en su nombre es para evitar repeticiones con los nombres de otras clases presentes en el proyecto global que impedirían su compilación. De forma contraria, existe la posibilidad de que otros módulos desarrollados por otros compañeros utilicen los mismos *assets* en sus trabajos. En este caso la solución es añadir una sola copia de esos componentes a un directorio de *assets* globales que utilicen ambos juegos.

Para seleccionar el acceso a un juego u otro de entre los módulos integrados en "El Planeta Sonoa", se dispone de un menú en el que primero puedes iniciar sesión con tu usuario de paciente o jugar sin usuario si no dispones de uno (Figura 46). A continuación, el juego consta de un menú de selección en el que puedes rotar el planeta para seleccionar los diferentes continentes y viajar a ellos para comenzar cada juego (Figura 47).

Figura 46: Pantalla de inicio de sesión de usuario en "El Planeta Sonoa".

Figura 47: Selección del juego "Tropikus" en "El Planeta Sonoa".

12.1 Ajustes desde la web

El apartado de ajustes realizado desde la página web de “El Planeta Sonoa” (Figura 48) funciona de manera análoga al menú *offline* detallado en el capítulo 7, sin embargo, este es totalmente transparente al jugador. De esta forma, es el terapeuta quien realiza los ajustes en la web y, cuando el jugador inicia el juego con su usuario y clave registrada en el servidor y elige a continuación viajar a la isla “Tropikus”, este módulo comienza automáticamente en la escena principal de juego. Esto es beneficioso para la gamificación del ejercicio, pues el jugador no será consciente de que su experiencia de juego ha sido previamente ajustada para sus necesidades.

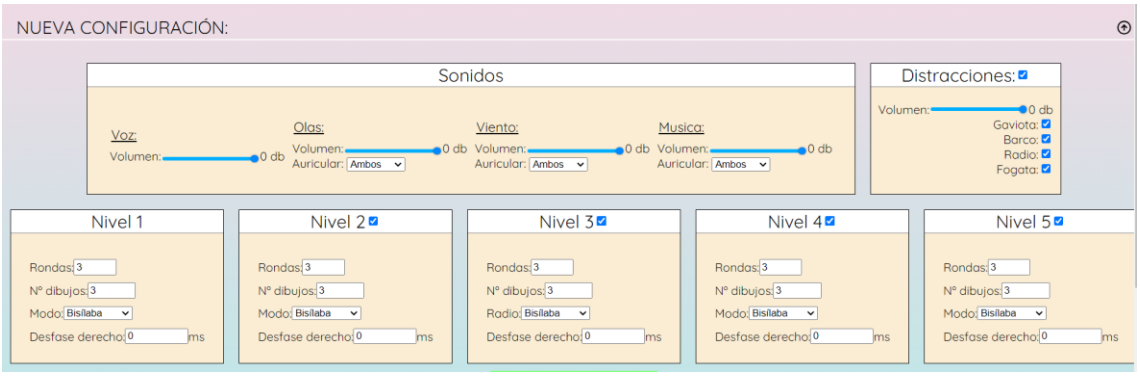


Figura 48: Pantalla de ajustes desde la página web.

Al comenzar el juego, este detecta si se está jugando en modo *offline* u *online* para cargar los datos. En el caso *offline* se muestra el menú de ajustes local y se juega con lo que en él se seleccione. En el caso *online* este menú no aparece y los ajustes realizados en la web son descargados en forma de fichero .json (Figura 49).

TROPI.json*

```
{"Configuration_id":"296",
"Voz_Volumen":"0",
"Ambiente_Volumen":"0", "Ambiente_Auricular":"0",
"Viento_Volumen":"0", "Viento_Auricular":"0",
"Musica_Volumen":"0", "Musica_Auricular":"0",
"Distracciones":"true", "Distracciones_Volumen":"0",
"D1":"true","D2":"true","D3":"true","D4":"true",
"NIVEL1":"true","N1_Rondas":"3","N1_Dibujos":"4","N1_Modo":"monosilaba","N1_Desfase":"-36",
"NIVEL2":"true","N2_Rondas":"3","N2_Dibujos":"6","N2_Modo":"trisilaba","N2_Desfase":"13",
"NIVEL3":"true","N3_Rondas":"3","N3_Dibujos":"3","N3_Modo":"trisilaba","N3_Desfase":"0"}
```

Figura 49: Fichero .json con ajustes descargados de la web.

El fichero .json almacena los ajustes que se corresponden a los realizados en el menú de la página web para poder ser leído por el juego. Además, este fichero queda guardado en la memoria hasta que se descarguen nuevos ajustes, de esta forma es posible jugar con los últimos ajustes descargados, aunque no haya conexión a internet.

Puede haber varios casos en los que se utiliza un tipo de ajustes u otro:

- Ajustes web (*online*):
  - Juego con usuario y conexión a la red, descargando la configuración aplicada.
  - Juego con usuario sin conexión a la red, cargando datos de ajuste de una sesión previa guardados (Figura 50).

Figura 50: Aviso de conexión fallida con datos guardados.

- Ajustes locales (*offline*):
  - Juego sin usuario.
  - Juego con usuario, sin conexión a la red y sin datos de ajuste previos guardados
  - Juego con usuario y conexión a la red, si no se han realizado ajustes para ese módulo en la web.

En ambos casos los ajustes se almacenan en el juego haciendo uso de la clase `TROPI_GameSettings()` para que no haya incompatibilidades independientemente del modo de funcionamiento. Para el caso *online* la clase forma parte de un objeto denominado `WebSettings` presente en el juego global y que almacena los ajustes de cada uno de los juegos añadidos desde la web para el usuario en uso. En el caso *offline*, la clase forma parte de un objeto específico para esta finalidad con nombre `Settings` que se encuentra presente en la escena de menú y se mantiene como objeto persistente en la escena de juego con `DontDestroyOnLoad()`.

Al principio, este paso de la integración ha sido causa de un conflicto en el momento en el que el juego debía buscar el objeto que contiene los ajustes en modo *offline*, debido a que el objeto `WebSettings` sigue presente en el juego, pero no contiene datos, es decir, existen dos objetos que contienen la clase `TROPI_GameSettings()`. Se añade una comprobación previa para identificar correctamente de qué clase deben obtenerse los datos de ajustes en cada caso.

12.2 Guardado de resultados

El juego guarda los resultados que se van obteniendo en el proceso de juego en la clase `TROPI_ResultsManager()`, que contiene variables para almacenar en forma de lista los resultados de cada ronda y nivel de juego. Contiene funciones para guardar datos, que una vez llamadas comprueban los ajustes y resultados en ese momento y los añade en una nueva entrada de la lista.

En el caso de “Tropikus”, se decide que los valores relevantes para su guardado son:

- Horas de inicio y final de ronda, en *strings* con formato HH:mm:ss.
- Las dos palabras correctas de cada ronda, en forma de *string* siendo primero la palabra del oído izquierdo y después la del derecho
- *Strings* con la cadena de dibujos excavados por el jugador en cada ronda, en orden. Para esto se requiere una función auxiliar de nombre `SaveWord()` que es llamada cada vez que el jugador excava un dibujo.
- Número de peticiones de volver a escuchar las palabras correctas en cada ronda
- Retardo aplicado en las palabras escuchadas en la ronda, en ms.
- Número de sílabas de las palabras del nivel.
- Fecha, en *string* con formato “MMMM, yyyy".

Cada vez que una ronda es finalizada se añaden todos estos valores a sus correspondientes listas para almacenarlos a excepción de la fecha, las sílabas y el retardo, que son valores constantes para todo el nivel. Estos resultados se separan en listas por niveles para diferenciar los resultados de cada nivel. Por último, al finalizar el juego (ya sea por haberlo completado o por interrumpirlo y salir desde el menú de pausa) se exportan todos los datos recopilados a la base de datos.

12.3 Resultados tras una sesión

Los resultados extraídos de una sesión de juego se almacenan en la base de datos de la página web de “El Planeta Sonoa” para su consulta. En ellos, es posible observar las palabras que el jugador a escogido en orden, viendo cuáles eran correctas, el número de repeticiones necesitadas y el tiempo que se ha empleado (Figura 51).

Resultados:

ID	Nivel	Ronda	Cadena correcta	Cadena resultado	Repeticiones	Tiempo
223	1	1	Mano Perro	Perro Mano	1	86s
223	4	2	Uvas Perro	Limón Perro Uvas	2	5s
223	4	1	Balón Reloj	Reloj Balón	0	5s
223	3	1	Pájaro Caballo	Caballo Pájaro	2	7s
223	2	1	Seis Flan	Seis Mar Flan	1	9s
223	4	3	Reloj Pluma	Reloj Pluma	0	8s
222	1	1	Sartén Mesa	Sartén Mesa	7	21s

Figura 51: Resultados de una sesión mostrados en la web.

Estos resultados pueden exportarse a un fichero de Microsoft Excel para trabajar con ellos. Además, se prevé que la página web muestre gráficas para organizar los resultados de interés del paciente. Esta funcionalidad se encuentra implementada en otros módulos de “El Planeta Sonoa” y en desarrollo para “Tropikus” por parte de Miguel Jiménez, técnico del GAMMA.

13. Impacto del proyecto

Este proyecto puede tener impacto en diversos aspectos, siendo el más importante de ellos su impacto social. Relacionando los diferentes impactos con los ODS (objetivos de desarrollo sostenible) [22], se deduce que el principal de ellos es el ODS número 3: salud y bienestar.

El objetivo general del proyecto se centra en el tratamiento del DPAC, creando materiales para el entrenamiento de diferentes ejercicios relacionados con ello. Este proyecto acerca y facilita la realización de estos ejercicios a los pacientes a través de videojuegos y puede tener un impacto muy beneficioso para su salud, facilitando el trabajo de los terapeutas que disponen de esta herramienta.

De igual forma y como resultado de estos beneficios para la salud y el ejercicio para el tratamiento de este desorden, el proyecto se relaciona también con el ODS número 4: educación de calidad, no sólo por la herramienta educativa que supone en sí mismo sino por facilitar también el acceso y entendimiento de otros recursos educativos.

Una herramienta que acerca estos ejercicios a todos los pacientes, incluso en sus casas, con pocos requisitos para su instalación y funcionamiento, supone un impacto muy positivo en el ODS número 10: reducción de las desigualdades. Este proyecto pone a disposición de forma libre la realización de ejercicios para el tratamiento del DPAC independientemente de la situación socioeconómica del paciente.

Por supuesto, el proceso de creación de un videojuego es una solución de ingeniería, siendo necesaria la innovación para encontrar la solución a las necesidades del proyecto haciendo uso de las últimas tecnologías disponibles para ello. Por esta razón, el proyecto se encuentra muy ligado al ODS número 9: industria, innovación e infraestructura. Gracias a tratarse de un recurso digital, su distribución y uso es muy sencilla y al alcance de los usuarios.

Por último, y como se ha mencionado a lo largo de toda la memoria, este es un proyecto realizado en equipo y en constante contribución entre los integrantes del GAMMA y las especialistas del centro de audiología Aurea TAV. Un trabajo en cooperación de estas características se identifica con el ODS número 17: alianzas para lograr los objetivos. Estas asociaciones inclusivas pueden darse a cualquier nivel (mundial, regional, nacional y local) y su objetivo es establecer una visión y unos objetivos compartidos que se centren primero en las personas y el planeta.

Así, se valora como positivo el impacto que un proyecto como este puede tener en la sociedad y se pone en valor continuar trabajando para su realización y distribución a aquellos que lo necesiten.

14. Conclusiones

Como cierre a esta memoria sobre este proyecto de fin de grado, se va a resumir el trabajo realizado, así como los resultados obtenidos y las dificultades e imprevistos hallados en el proceso de alcanzar los objetivos marcados.

Se alcanza el primer objetivo previsto, generando la lógica interna capaz de realizar la distribución aleatoria de dibujos que excavar en la playa. En este proceso surgen nuevos requisitos e indicaciones que alargan el tiempo de trabajo necesario, como la aparición en una zona aleatoria de la isla de la cadena de dibujos y la opción de aplicar retardo entre ambas palabras escuchadas. Esto pone en valor la importancia de prever los posibles cambios y reajustes que pueden verse necesarios en el desarrollo de una tarea a la hora de estimar el tiempo necesario para ello, siendo necesario contar con un posible margen para que el calendario sea flexible.

El trabajo continúa completando el objetivo de la inclusión de sonidos, objetos y modelos animados en el videojuego para terminar de crear la ambientación de este. En este apartado se destaca la importancia de la reutilización y modificación de recursos preexistentes. Gracias a que existen materiales creados con anterioridad es posible reducir el tiempo dedicado a esta tarea, que en muchas ocasiones escapa de las capacidades de carácter artístico requeridas en un grado de Ingeniería.

Para finalizar el desarrollo individual de la secuencia de juego se procede a la creación de cinemáticas y todas las casuísticas posibles que pueden darse en su aparición, que era otro de los objetivos. Este es un apartado de gran complejidad para el que resulta necesaria una inversión de tiempo mayor a la prevista, y pone en valor el trabajo de realización de pruebas del juego para encontrar cada posible acción que desencadene una situación no deseada. Se han realizado animaciones y cinemáticas sencillas, quedando margen para añadir nuevas cinemáticas y mejorar sus animaciones, como se detalla en el capítulo 15.

Con el módulo ya desarrollado, se une el trabajo hecho por los miembros del equipo alcanzando el objetivo de la integración en el proyecto global y página web. Ya que se está combinando el trabajo realizado por diferentes personas, es en este paso donde se aprecia la importancia de utilizar versiones de software comunes para evitar incompatibilidades, especialmente en Unity.

En general, el desarrollo de un videojuego de estas características muestra el valor de la comunicación entre miembros del equipo y expertos en cada materia, como en este caso las audiólogas del centro Aurea TAV. También se deben tener en cuenta los posibles perfiles que puede tener un jugador que haga uso del juego, pues cada experiencia puede ser muy distinta. Aquí reside la importancia de explicar los diferentes controles y acciones que puede realizar el jugador y cómo debe realizarlas, así como la creación de un manual para resolver dudas. Esta labor ha sido realizada, alcanzando así el último de los objetivos marcados, a excepción de la elaboración de un GDD que será redactado posteriormente a la publicación del proyecto.

Se trata de un proyecto ambicioso, con secciones de trabajo muy diferenciadas que habitualmente no se realizan por una sola persona. A pesar de haberse necesitado más tiempo del estimado inicialmente o haber necesitado aplicar cambios en el proceso, el resultado obtenido ha sido satisfactorio y acorde a los objetivos y requisitos principales. No obstante, queda margen para posibles mejoras futuras que añadir.

15. Futuras mejoras del proyecto

"El Planeta Sonoa" es un proyecto muy ambicioso, incluso cada uno de sus módulos lo es, por ello es inabarcable en el tiempo asignado a un proyecto de fin de grado estipulado con 12 ECTS (300 horas de trabajo). Esto hace que existan diversas características o mejoras que no han sido implementadas por esta falta de recursos al haberse priorizado otras.

En el caso de "Tropikus", la principal implementación que puede ser deseable en el futuro es la adición de más palabras al banco de *prefabs* disponible del videojuego, para hacerlo más variado y rico. El código del juego se encuentra diseñado para funcionar con la posibilidad de añadir más palabras en el futuro realizando cambios mínimos en el mismo. El proceso de añadir nuevas palabras puede resumirse en la creación de nuevos *prefabs* similares a los ya existentes y añadirlos al *script* `TROPI_GenerateDrawings.cs`. Como se trata de una tarea importante, se detallan estos pasos con más atención en el Anexo III. De forma similar, también es posible añadir nuevas decoraciones desbloqueables a la isla de forma sencilla, detallado en el Anexo IV.

Existen algunas mejoras gráficas y estéticas generales que pueden añadirse al juego para hacerlo más llamativo al jugador:

- Ampliación y mejora en las animaciones de los guacamayos para aportar más variedad y animaciones más logradas, además de nuevos sonidos asociados a estas.
- Adición de nuevos modelos animados similares a las gaviotas que aparecen volando para dar más vida al entorno.
- Creación de una animación mejorada de excavación, utilizando los modelos de pala y llaves ya incluidos en el proyecto, pero finalmente no utilizados.
- Creación de nuevas cinemáticas con más detalles y posiciones de cámara. Existe la posibilidad de crear cinemáticas navegando para llegar o salir de la isla en barco.
- Integración de nuevos sonidos de fondo.

Hay dos funcionalidades del juego que no han sido implementadas, pero sí fueron planteadas en un inicio. Estas han sido descartadas debido a su complejidad de implementación, de modo que, a diferencia de las mejoras detalladas anteriormente, estas pueden resultar más costosas:

- Selector de voz: añadir un nuevo ajuste que permita escoger diferentes voces para el juego. Esto aporta una nueva dimensión de estudio en los pacientes, que pueden responder de forma diferente en función de la voz escuchada. Sin embargo, su implementación requiere la creación de nuevos doblajes y multiplica la cantidad de archivos de audio presentes en el juego (que ya son numerosos debido al número de palabras implementadas).
- Modo de juego de texto: añadir un segundo modo de juego en el que el jugador no escucha dos palabras en cada oído, sino una palabra en uno de los oídos (de forma ajustable) y una narración más extensa relacionada con la palabra escuchada en el otro oído. Esta es otra prueba habitual en el ámbito de la audiología para los estudios de escucha dicótica, centrados en la capacidad de atención del paciente. El funcionamiento del juego sería

similar, necesitando excavar únicamente un dibujo correcto en este caso. El código principal del juego presente en `PlayAgent` ya contempla una variación en el número de dibujos que se deben excavar a través de la variable `max_round_keys`. Así, la implementación de nuevos modos de juego depende únicamente del *script* `TROPI_GenerateDrawings.cs` y de los ajustes. Al igual que con la mejora anterior, esto hace necesaria la creación de nuevos doblajes mucho más extensos.

Más allá de las mejoras correspondientes al propio módulo de "Tropikus", existen consideraciones a tener en cuenta de cara al proyecto global y el resto de los módulos. Es recomendable establecer una estética común, tanto en la ambientación como en la interfaz de usuario, para dar una mejor sensación de unidad y relación entre los módulos. Los ajustes y las teclas escogidas para el manejo de los juegos son otros aspectos que deben funcionar de manera idéntica entre juegos para evitar confusiones en los jugadores.

Durante el tiempo de desarrollo de este proyecto se han tenido en cuenta la mayoría de estas consideraciones con los demás compañeros que han trabajado simultáneamente, pero deben mantenerse para futuros módulos e intentar añadirse también en los módulos desarrollados anteriormente y el juego central.

16. Referencias

[1] AEVI, «Anuario 2022,» 2022.

[2] I. García Lázaro, «Universidad Isabel I,» 13 abril 2021. [En línea]. Available: https://www.ui1.es/blog-ui1/quienes-son-los-nativos-digitales. [Último acceso: 11 julio 2023].

[3] «Ventajas de la gamificación en el aula: qué herramientas utilizar y cómo aplicarla,» 28 febrero 2022. [En línea]. Available: https://www.becas-santander.com/es/blog/gamificacion-en-el-aula.html. [Último acceso: 11 julio 2023].

[4] «Unity,» [En línea]. Available: https://unity.com/es. [Último acceso: 11 julio 2023].

[5] «Blender,» [En línea]. Available: https://www.blender.org/. [Último acceso: 11 julio 2023].

[6] M. Eckert et al., «The Blexer system – Adaptive full play therapeutic exergames with web-based supervision for people with motor dysfunctionalities,» *EAI Endorsed Transactions on Serious Games,* vol. 5, nº 16, 2018.

[7] O. Cañete, «Desorden del procesamiento auditivo central (DPAC),» *Rev. Otorrinolaringol. Cir. Cabeza Cuello,* vol. 66, pp. 263-273, 2006.

[8] «Aurea TAV,» [En línea]. Available: https://www.aureatav.es/. [Último acceso: 11 julio 2023].

[9] L. Jiménez Sampedro, «Diseño e implementación de un videojuego terapéutico para problemas asociados con el déficit de atención.,» Madrid, 2018.

[10] E. García Medina, «Memoria final de alumno en prácticas,» Madrid, 2022.

[11] M. C. Egocheaga Ojeda, «Desarrollo e implementación de la segunda fase del videojuego terapéutico "El Planeta de Amalia",» Madrid, 2021.

[12] B. Morar, «Ampliación de un videojuego terapéutico para mejora del procesamiento auditivo: creación de un módulo de historias.,» Madrid, 2023.

[13] M. Ortega García, «Ampliación de un videojuego terapéutico para mejora del procesamiento auditivo: desarrollo de un módulo de secuencias melódicas,» Madrid, 2023.

[14] M. Li, «Ampliación de un videojuego terapéutico para mejora del procesamiento auditivo: creación de un módulo de localización de sonidos 3D.,» Madrid, Actualmente en elaboración. Fecha prevista: 2023.

[15] A. T. Páez Pinilla, «Batería de evaluación del procesamiento auditivo dicótico (BEPADI),» Bogotá, 2000.

[16] Asana, «Cómo utilizar el método de la ruta crítica en la gestión de proyectos,» 16 octubre 2021. [En línea]. Available: https://asana.com/es/resources/critical-path-method. [Último acceso: 12 julio 2023].

[17] E. Dogan, «Low Poly Water | Particles/Effects | Unity Asset Store,» [En línea]. Available: https://assetstore.unity.com/packages/tools/particles-effects/lowpoly-water-107563. [Último acceso: 10 Julio 2023].

[18] P. Lucas Olivares, I. Sáez Elvira y L. García Várez, «Colorless,» 2021.

[19] V. Betoret Ferrero, «Seagulls animated,» 6 diciembre 2020. [En línea]. Available: https://sketchfab.com/3d-models/seagulls-animated-73aed843190a4dfda55f2b65cc0f8d63. [Último acceso: 11 julio 2023].

[20] solvkart, «Turbosquid,» 28 julio 2020. [En línea]. Available: https://www.turbosquid.com/es/3d-models/stylized-chest-animations-3d-model-1597758#. [Último acceso: 12 julio 2023].

[21] Alien Interactive, «Turbosquid,» [En línea]. Available: https://www.turbosquid.com/es/3d-models/3d-bird-parrot-blue-ar-1301496. [Último acceso: 11 julio 2023].

[22] Naciones Unidas, «Objetivos de Desarrollo Sostenible,» [En línea]. Available: https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/. [Último acceso: 11 julio 2023].

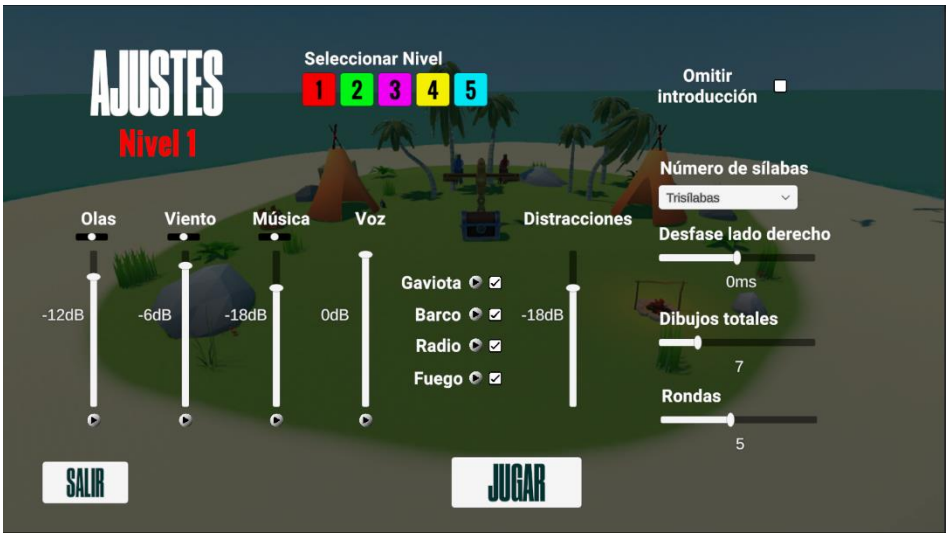
ANEXO I: Manual de juego

1. Manual detallado

Configuración local jugando sin usuario

Si ha entrado con nombre de usuario y contraseña, puede saltar al apartado 2. Si se ha elegido "Continuar sin usuario", se dispone de una pantalla de ajustes previa (Figura 1).

Estas pantallas solamente son accesibles jugando en el modo "sin usuario", y sirven para hacer pruebas de ajuste inmediatos. Los ajustes de los pacientes se hacen online vía la página web http://sonoa.citsem.upm.es/.



Anexo I. Figura 1: Pantalla de ajustes.

En estos ajustes es posible ajustar el nivel y la panoramización de los distintos sonidos presentes en la escena. Estos sonidos son la voz, olas, viento y música. Además, existen otros ruidos llamados "distracciones" (gaviota, barco, radio y fogata) que suenan en espacialmente en función de la posición del jugador, es posible activar y desactivar y controlar su nivel general. Reproduciendo los sonidos, se puede realizar una pre-escucha de la mezcla final.

Controles

- Avanzar hacia delante: letra "W" en teclado o joystick izquierdo a arriba en mando.
- Avanzar hacia atrás: letra "S" en teclado o joystick izquierdo a abajo en mando.
- Avanzar hacia la izquierda: letra "A" en teclado o joystick izquierdo a la izquierda en mando.
- Avanzar hacia la derecha: letra "D" en teclado o joystick izquierdo a la derecha en mando.
- Movimientos de cámara (vista del jugador): movimiento del ratón o movimiento del joystick derecho.
- Excavar, comenzar rondas, omitir intro: letra "E" en teclado o botón "A" en mando.
- Repetir las palabras: letra "R" en teclado o botón "Y" en mando.
- Salto: barra espaciadora en teclado o botón "LB" en mando
- Correr: mantener pulsado mayúsculas izquierda (*shift* izquierdo) en teclado o mantener botón "RB" en mando

Proceso de juego

El jugador aparece en el puerto de la isla. Al acercarse al centro de la isla, los guacamayos le piden su ayuda y comienza el juego. Terminada la explicación, el jugador debe comenzar la ronda y escuchar con atención las palabras que le dicen los guacamayos (Figura 2).

Anexo I. Figura 2: Inicio del juego.

En ese momento, tiene que moverse por la playa para buscar los dibujos (cada vez aparecen en un lugar diferente). Entre los dibujos, dos se corresponden con las palabras escuchadas y son los correctos que deben excavarse. En caso de olvidar las palabras, pueden volverse a escuchar pulsando la tecla de repetición (“R” en teclado y botón “Y” en mando). Para excavar los dibujos, se debe estar cerca de ellos. En el momento en que un dibujo pueda excavarse aparece encima de él el botón que debe pulsarse para excavar (“E” en teclado y botón “A” en mando). Si el dibujo es correcto, se suma una llave (Figura 3).

Anexo I. Figura 3: Proceso de excavar dibujos.

Al finalizar el nivel completando todas las rondas (puede verse el progreso de las rondas en la esquina superior derecha) se logra abrir el cofre si se han conseguido las llaves suficientes, obteniendo un objeto de regalo (Figura 4). A continuación, comienza un nuevo nivel o finaliza el juego si no hay más niveles activados.

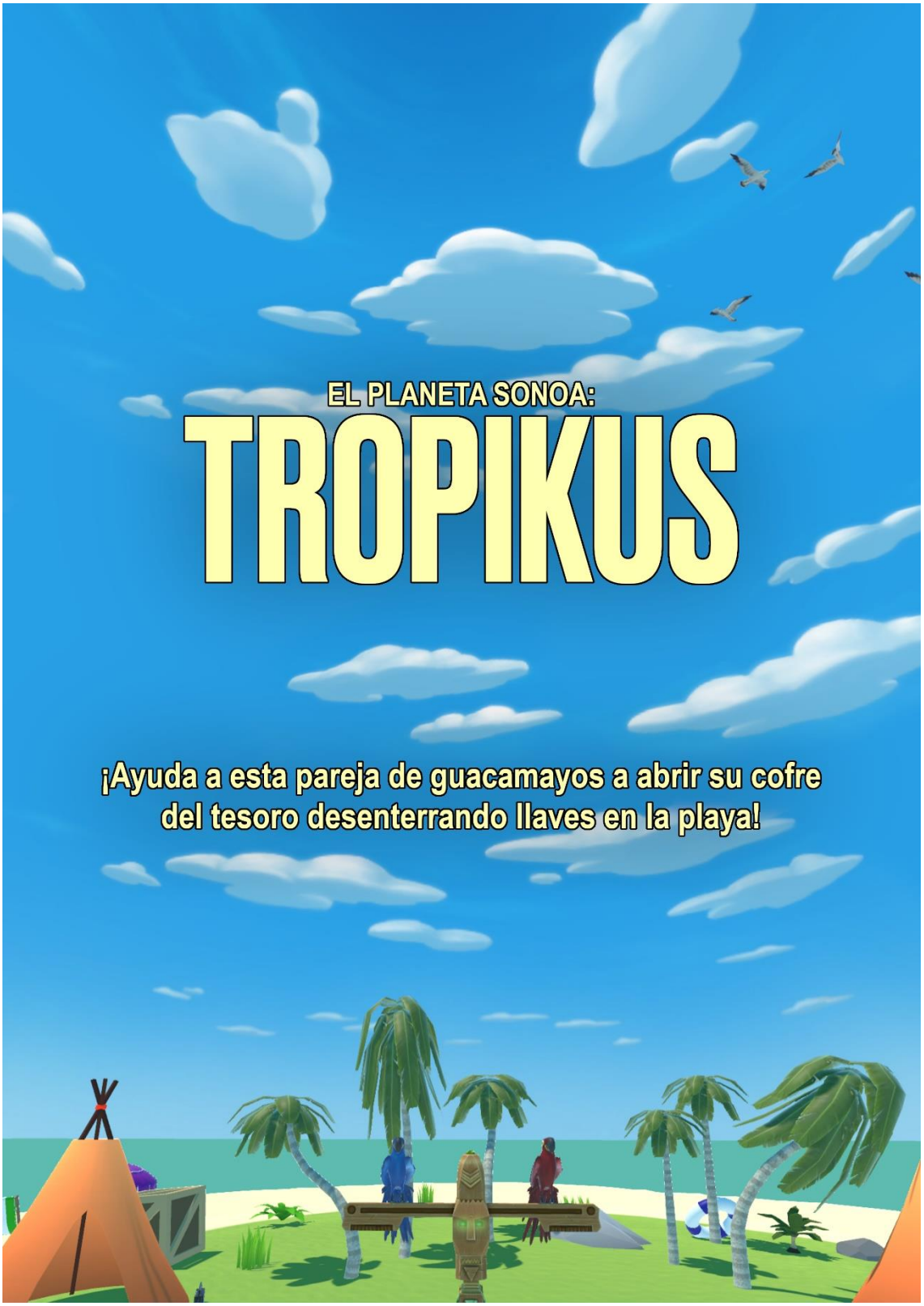
Anexo I. Figura 4: Final de nivel.

Salir del juego

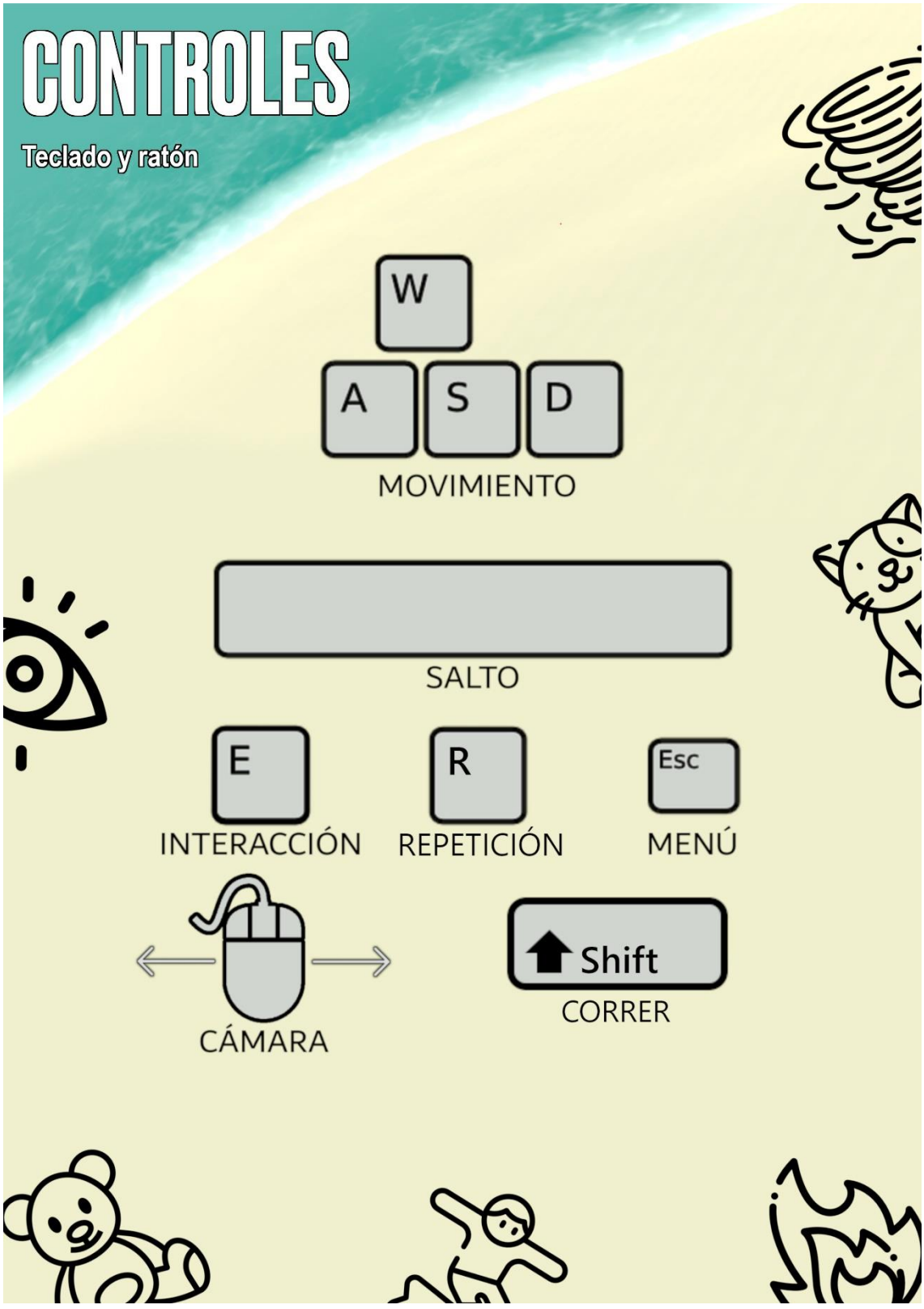
Se puede salir del juego en cualquier momento pulsando la tecla Escape en teclado o Menú en mando, que activa el menú de pausa desde el que salir del juego. En la pantalla de ajustes existe directamente un botón para salir en la esquina inferior izquierda del menú. Si el juego se ha finalizado correctamente, también aparece un botón para salir del juego en la pantalla final (Figura 5).

Anexo I. Figura 5: Izquierda: pantalla de pausa. Derecha: pantalla final.

2. Manual resumido para niños



Anexo I. Figura 6: Portada de manual resumido.



Anexo I. Figura 7: Controles para teclado y ratón de manual resumido.

Anexo I. Figura 8: Controles para mando de manual resumido.

ANEXO II: Proceso de grabación y voz sintética

Dada la importancia de la escucha en el videojuego es necesario realizar una grabación de calidad para el doblaje de los personajes del juego. Para ello se ha hecho uso de material propio y software de DAW (*Digital Audio Workstation*) como Reaper [1].

Se ha utilizado un micrófono de calidad profesional Shure sm58, micrófono dinámico utilizado ampliamente para la microfonía de la voz en sistemas de grabación o de refuerzo sonoro.

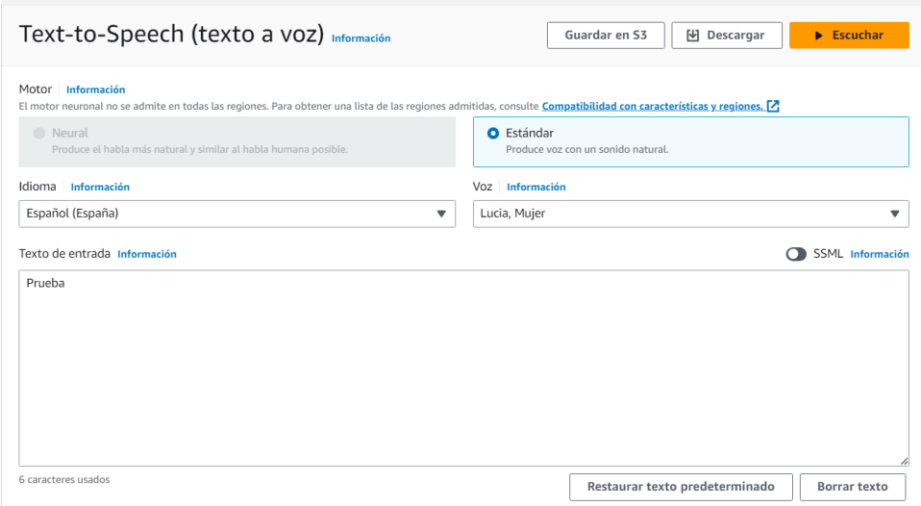
El conexionado del micrófono y procesamiento de la señal de audio se realiza a través de la tarjeta de sonido externa Focusrite Scarlett 2i2 de segunda generación, junto con la instalación de sus drivers correspondientes para su correcto funcionamiento en el ordenador (Asus Strix GL502VMK) con Reaper (Figura 1).

Anexo II. Figura 1: Diagrama de conexionado para grabación de voz.

Debido a que el videojuego combina doblaje grabado y voz sintética femenina, y con el objetivo de que ambas voces se parezcan para mayor coherencia, los doblajes han sido grabados con la colaboración de María Pérez Casero. De esta forma, las líneas de diálogo de los guacamayos grabadas con voz real son:

- ¡Qué bien que has venido! Necesitamos tu ayuda para abrir este cofre desenterrando sus llaves en la playa.
- Escucharás dos palabras y tendrás que excavar en los dibujos, como estos de aquí, que se correspondan con ellas.
- ¿Todo listo para escuchar las siguientes palabras?
- ¡Muy bien!
- No, así no…
- ¡Genial! Hemos recuperado un nuevo objeto.
- Vaya, parece que esta vez no has encontrado suficientes llaves.
- ¡Muchas gracias por tu ayuda!
- Esto es un ejemplo del volumen de la voz en el juego.

Con el objetivo de poder añadir palabras en cualquier momento, que deben ser siempre con la misma voz, así como de que se escuchen con claridad y pronunciación neutra, se utiliza una voz sintética para la lectura de las palabras. El generador de voz sintética utilizado en este proyecto es Amazon Polly [2] (Figura 2), uno de los servicios ofrecidos por AWS (Amazon Web Services). Si bien este servicio cuenta con un servicio de pago, es posible generar de forma gratuita textos de pocos caracteres. Se selecciona la voz de mujer, Lucía.



Anexo II. Figura 2: Generación de palabras con voz sintética en Amazon Polly.

Los audios generados hasta el momento constan de una edición muy sencilla que debe replicarse en caso de añadir nuevas palabras. Concretamente, los audios generados y descargados de Amazon Polly han sido recortados para reducir partes en silencio en el audio antes y después de su reproducción, lo que se traduciría en el juego como un retardo no deseado. Además, el audio original descargado debe ver aumentada su ganancia en 6dB. Siguiendo este procedimiento se ha generado la lista de palabras presente en el videojuego (Tabla 1).

El generador de texto a voz de Amazon Polly aún no cuenta con la funcionalidad de generar la voz neural en España, por lo que se utiliza el formato estándar, siendo posible reemplazar estos audios por otros nuevos haciendo uso de la versión neural para obtener resultados más realistas cuando esta funcionalidad se encuentre disponible.

La diferencia entre ambos métodos, explicada en la información de AWS, depende del método de generación. Las voces estándar utilizan síntesis concatenativa, método que une los fonemas del habla grabada, produciendo un habla sintetizada muy natural pero incapaz de reproducir las variaciones en el habla. El sistema neural de Amazon Polly no utiliza síntesis concatenativa estándar para producir habla. Tiene dos partes:

- Una red neuronal que convierte una secuencia de fonemas, que son las unidades básicas del lenguaje, en una secuencia de espectrogramas, que son capturas de los niveles de energía en diferentes bandas de frecuencia.
- Un *vocoder* (analizador y sintetizador de voz), que convierte los espectrogramas en una señal de audio continua.

Hacer uso de un generador de texto a voz de estas características aporta flexibilidad en el desarrollo del juego, siendo independiente de la persona que desarrolle nuevas palabras.

Anexo II. Tabla 1: Palabras generadas con voz sintética añadidas en el juego.

Índice	Monosílaba	Bisílaba	Trisílaba
1	Cruz	Árbol	Abeja
2	Flan	Barco	Abrazo
3	Luz	Carta	Anillo
4	Flor	Casa	Aplauso
5	Pan	Dado	Ballena
6	Sal	Fuego	Bellota
7	Sol	Balón	Caballo
8	Tos	Gato	Camello
9	Tren	Lápiz	Corbata
10	Dos	Mano	Cuadrado
11	Tres	Limón	Estrella
12	Seis	Luna	Fantasma
13	Mes	Perro	Gallina
14	Chef	Amor	Hospital
15	Gol	Nube	Jirafa
16	Golf	Ocho	Ladrillo
17	Surf	Piña	Antifaz
18	Nuez	Pluma	Peluche
19	Mar	Queso	Pájaro
20	Paz	Ratón	Patata
21	Pie	Reloj	Plátano
22	Pez	Sartén	Camisa
23	Rey	Mesa	Tornado
24	Té	Café	Ventana
25	Ver	Uvas	Zapato

Referencias

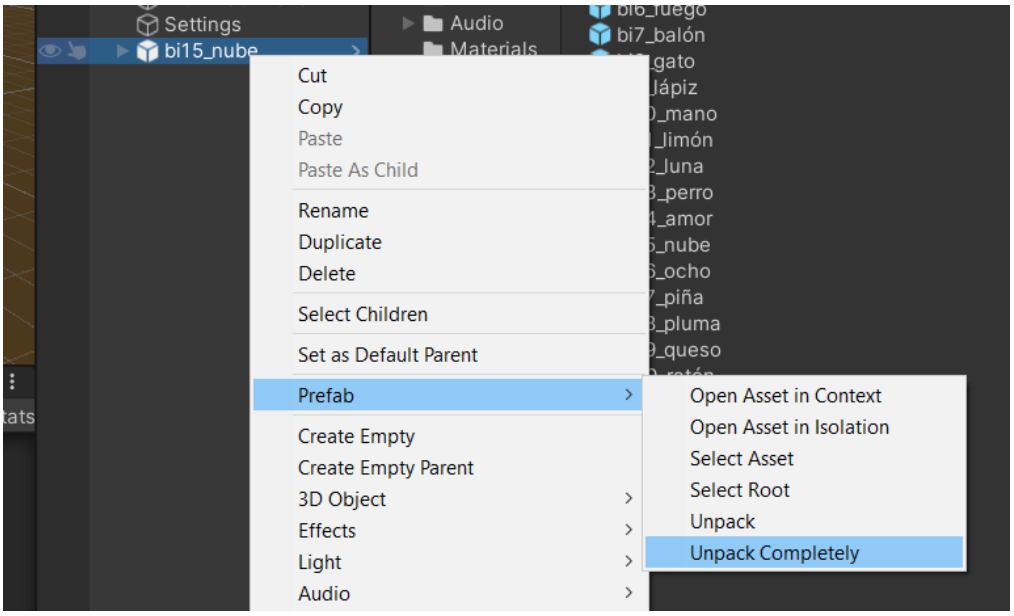
[1] «Reaper,» [En línea]. Available: https://www.reaper.fm/. [Último acceso: 11 julio 2023].

[2] «Amazon Polly,» [En línea]. Available: https://aws.amazon.com/es/Polly/. [Último acceso: 11 julio 2023].

ANEXO III: Guía de implementación de nuevas palabras

En el futuro, puede ser deseable aumentar el banco de palabras disponibles en el juego para tener una mayor variedad y posibilidades. Por esta razón se expone detalladamente, a modo de guía, el proceso que es necesario seguir para incluir nuevas palabras en el juego.

1. Tomar como referencia un *prefab* ya existente correspondiente a cualquiera de los dibujos anteriores, ubicados en el directorio \Assets\Prefabs. Para esto se añade a la escena uno de los *prefabs* y se selecciona la opción "*prefab* → *unpack completely*" para transformarlo en un nuevo objeto independiente del *prefab* original (Figura 1).

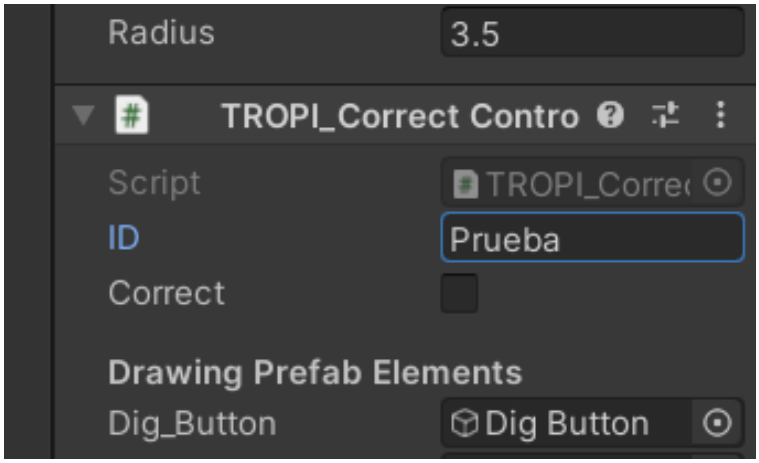


Anexo III. Figura 1: Desempaquetado de *prefab*.

2. Renombrar el objeto siguiendo la nomenclatura utilizada para los demás *prefabs*: añade un prefijo ("mono", "bi", "tri") en función del número de sílabas seguido de un número para ordenarlos y la palabra correspondiente. Por ejemplo, se va a renombrar el objeto a "`bi26_prueba`" (Figura 2).

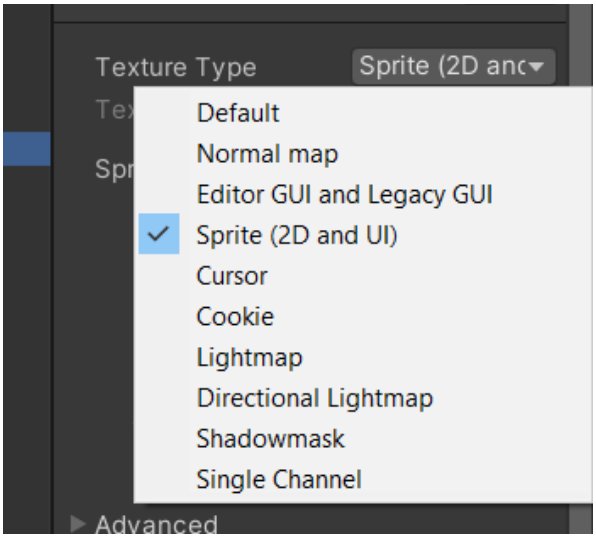
Anexo III. Figura 2: Cambio de nombre del objeto.

3. En la ventana de inspector del nuevo objeto se edita el campo `ID` escribiendo la nueva palabra (Figura 3). Este ID (identificador) será el utilizado para mostrar los resultados de la partida.



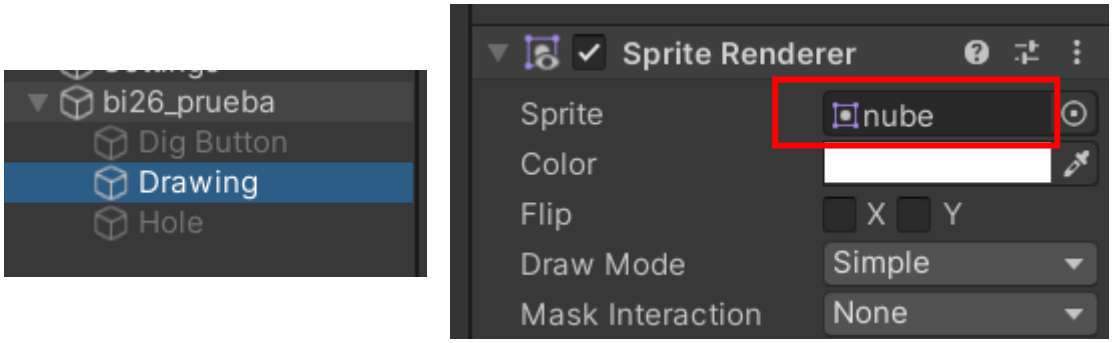
Anexo III. Figura 3: Definición del parámetro ID en el inspector.

4. Añadir un icono plano con trazo negro sobre fondo transparente en tamaño 512x512 en formato .png para crear estos *sprites*. Deben guardarse en la carpeta *Assets*\UI\Textures\ seleccionando el directorio correspondiente al número de sílabas de la palabra. En el caso de los *prefabs* ya creados se ha utilizado el repositorio gratuito de iconos Flaticon [1] para obtener iconos de estas características, por lo que se recomienda su uso. En el inspector, debe seleccionarse la imagen y marcarla como *sprite* (Figura 4).



Anexo III. Figura 4: Definición de imagen como *sprite*.

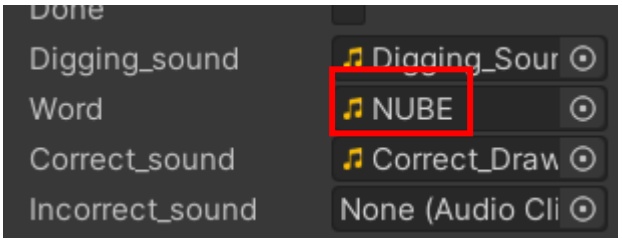
5. De nuevo en el objeto creado en la escena (llamado bi26_prueba en este caso), se despliega para ver los hijos añadidos al objeto y se selecciona el objeto "Drawing". En la variable `Sprite` del inspector se selecciona el nuevo *sprite* creado en el paso anterior (Figura 5).



Anexo III. Figura 5: Asignación del *sprite* en el inspector.

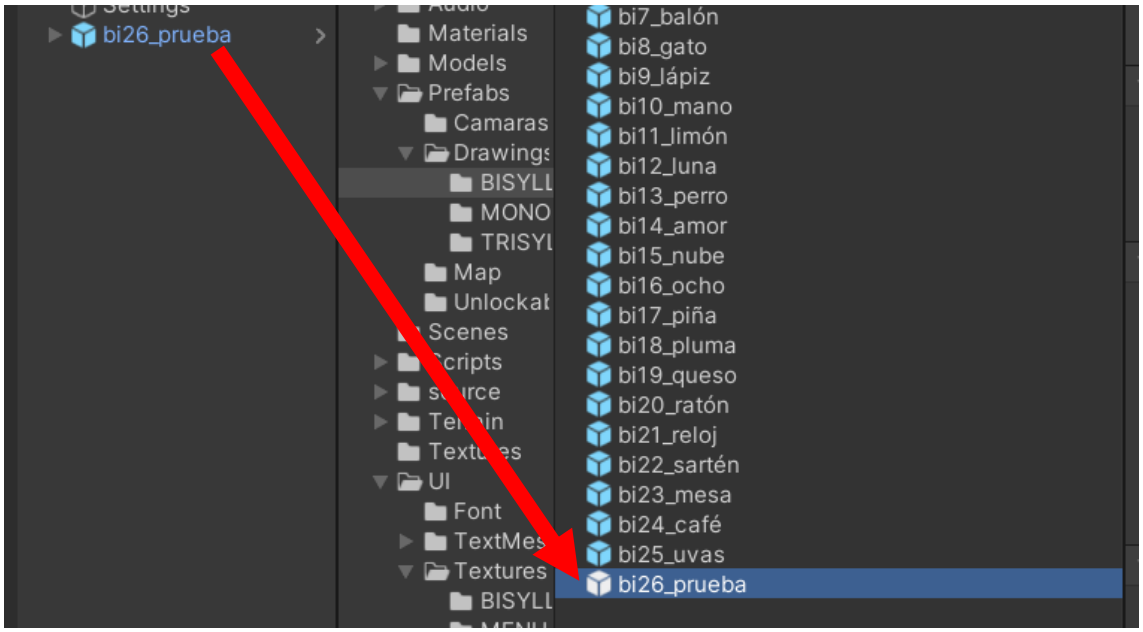
6. Al igual que con el icono, se añade un fichero de audio en formato .mp3 con la voz diciendo la palabra en la carpeta *\Assets*\Audio seleccionando el directorio correspondiente con el número de sílabas de la palabra. Este audio es generado con voz sintética como se indica en el Anexo II.

7. Se añade el clip de audio correspondiente a la palabra en el inspector del objeto (llamado en este caso `bi26_prueba`) en el espacio llamado "`Word`" (Figura 6).



Anexo III. Figura 6: Asignación del clip de audio de la palabra en el inspector.

8. Se genera un nuevo *prefab* a partir del objeto que se ha editado. Puede hacerse arrastrando el objeto desde la escena a la carpeta de *prefabs* correspondiente con el número de sílabas (Figura 7).



Anexo III. Figura 7: Creación del nuevo *prefab*.

9. Editar el código `TROPI_GenerateDrawings.cs` para añadir una referencia al nuevo *prefab* creado. En el apartado de variables se despliega la lista correspondiente a la instancia de los diferentes *prefabs* y se añade una nueva línea siguiendo el formato de las anteriores. En este caso se declara el objeto "bi26" (Figura 8).

```
public GameObject bi25;
public GameObject bi26;
#endregion
```

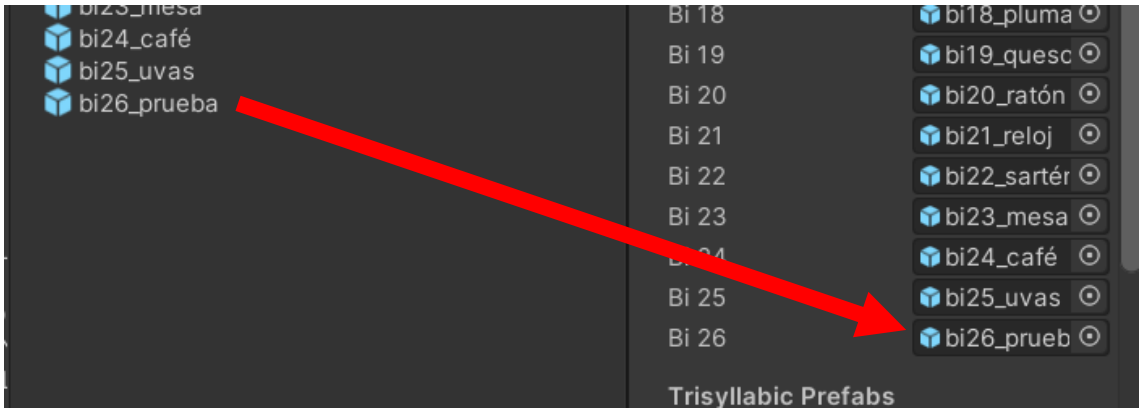
Anexo III. Figura 8: Definición de nuevo objeto.

10. Se sigue editando el código en el método `NewLevel(int syllable_number)`. Se amplía el tamaño del *array* para corresponderse con el número de *prefabs* creados de palabras con el número de sílabas correspondiente y se añade una nueva línea para añadir el *prefab* creado al *array* contenedor de todos los *prefabs* (Figura 9).

```
case 1:
    prefabsDrawings = new GameObject[25];
    prefabsDrawings[0] = bi1;
    prefabsDrawings[1] = bi2;
    prefabsDrawings[2] = bi3;
    prefabsDrawings[3] = bi4;
    prefabsDrawings[4] = bi5;
    prefabsDrawings[5] = bi6;
    prefabsDrawings[6] = bi7;
    prefabsDrawings[7] = bi8;
    prefabsDrawings[8] = bi9;
    prefabsDrawings[9] = bi10;
    prefabsDrawings[10] = bi11;
    prefabsDrawings[11] = bi12;
    prefabsDrawings[12] = bi13;
    prefabsDrawings[13] = bi14;
    prefabsDrawings[14] = bi15;
    prefabsDrawings[15] = bi16;
    prefabsDrawings[16] = bi17;
    prefabsDrawings[17] = bi18;
    prefabsDrawings[18] = bi19;
    prefabsDrawings[19] = bi20;
    prefabsDrawings[20] = bi21;
    prefabsDrawings[21] = bi22;
    prefabsDrawings[22] = bi23;
    prefabsDrawings[23] = bi24;
    prefabsDrawings[24] = bi25;
    prefabsDrawings[25] = bi26;
    break;
```

Anexo III. Figura 9: Edición del *array* de objetos.

11. En último lugar, en la escena de playa se busca el objeto `Generate Drawings` y se añade en el inspector el *prefab* correspondiente al nuevo dibujo. Con esto finaliza el proceso de adición de nuevas palabras en el juego (Figura 10).



Anexo III. Figura 10: Asignación del nuevo *prefab* a la clase `GenerateDrawings()` en el inspector.

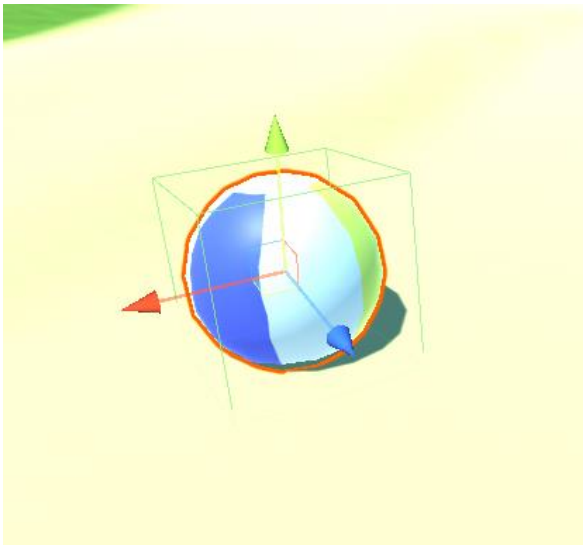
Referencias

[1] «flaticon,» [En línea]. Available: https://www.flaticon.es/. [Último acceso: 11 julio 2023].

ANEXO IV: Guía de implementación de nuevas decoraciones

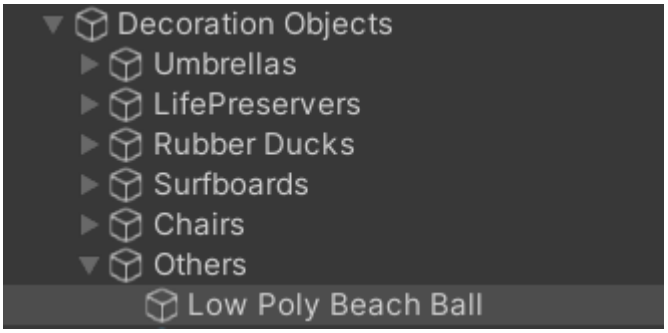
Para añadir nuevas decoraciones en el juego se debe seguir un procedimiento muy sencillo:

1. En primer lugar, se añade a la escena de playa el objeto que se quiera añadir como nuevo desbloqueable que debe incluir todos los elementos asociados a este (como un *collider*). Este objeto puede ser un recurso externo o creado propio, su proceso de obtención es independiente para su implementación como decoración (Figura 1).



Anexo IV. Figura 1: Ejemplo de objeto decorativo añadido a la escena, con *box collider*.

2. El objeto añadido en la escena debe ubicarse como hijo del objeto `Decoration Objects`, si bien esto no es necesario, facilita su organización (Figura 2).



Anexo IV. Figura 2: Objeto colocado como hijo de `Decoration Objects`.

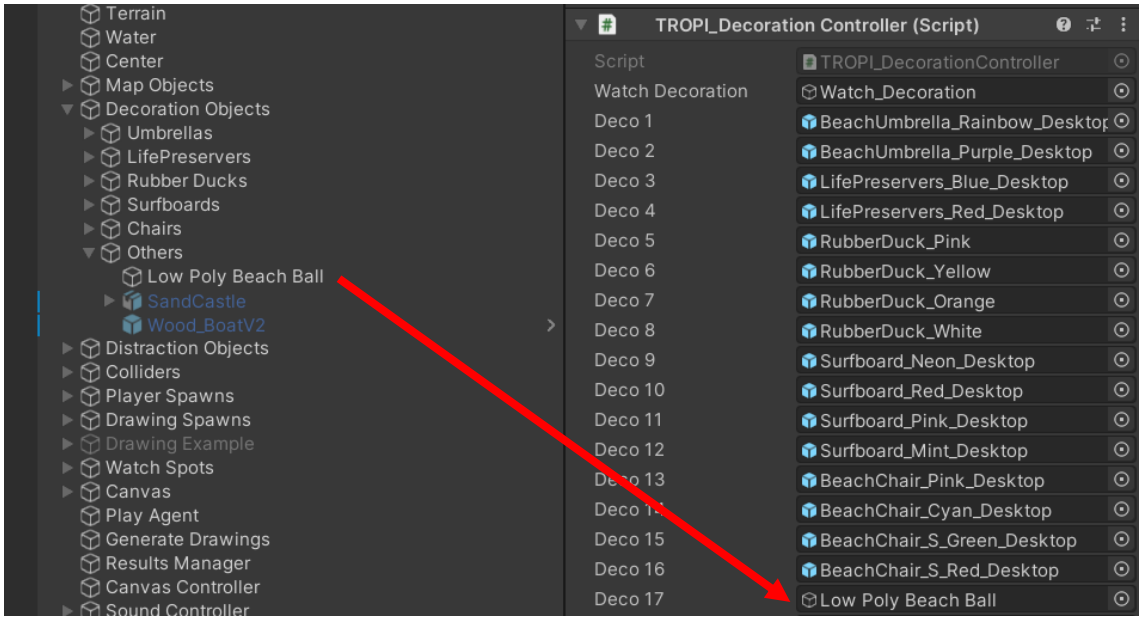
3. Se edita el *script* `TROPI_DecorationController.cs` para añadir una nueva definición de un objeto y su inclusión en el *array* de decoraciones, del cual también debe modificarse su tamaño para albergar un nuevo objeto más. Se muestra en la Figura 3 la definición para añadir un nuevo objeto número 19.

```
public GameObject deco1;
public GameObject deco2;
public GameObject deco3;
public GameObject deco4;
public GameObject deco5;
public GameObject deco6;
public GameObject deco7;
public GameObject deco8;
public GameObject deco9;
public GameObject deco10;
public GameObject deco11;
public GameObject deco12;
public GameObject deco13;
public GameObject deco14;
public GameObject deco15;
public GameObject deco16;
public GameObject deco17;
public GameObject deco18;
public GameObject deco19;
#endregion
#endregion
```

```
DecorationsArray = new GameObject[19];
DecorationsArray[0] = deco1;
DecorationsArray[1] = deco2;
DecorationsArray[2] = deco3;
DecorationsArray[3] = deco4;
DecorationsArray[4] = deco5;
DecorationsArray[5] = deco6;
DecorationsArray[6] = deco7;
DecorationsArray[7] = deco8;
DecorationsArray[8] = deco9;
DecorationsArray[9] = deco10;
DecorationsArray[10] = deco11;
DecorationsArray[11] = deco12;
DecorationsArray[12] = deco13;
DecorationsArray[13] = deco14;
DecorationsArray[14] = deco15;
DecorationsArray[15] = deco16;
DecorationsArray[16] = deco17;
DecorationsArray[17] = deco18;
DecorationsArray[18] = deco19;
```

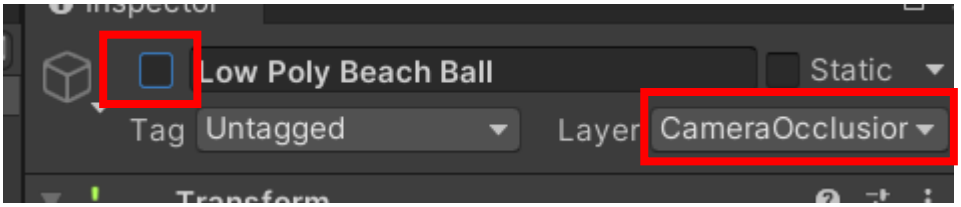
Anexo IV. Figura 3: Edición de código para añadir nueva definición de objeto.

4. En el inspector de la clase `TROPI_DecorationController()`, añadida en el objeto `Decoration Controller` de la escena, se añade la nueva decoración en su variable correspondiente (Figura 4).



Anexo IV. Figura 4: Asignación del objeto en el inspector.

5. Desactivar el objeto añadido en el inspector para que no se encuentre visible al inicio del juego y seleccionar la capa "CameraOcclusion", que evita que la cámara pueda traspasar objetos sólidos (Figura 5).



Anexo IV. Figura 5: Desactivación del objeto y selección de capa.

De esta forma finaliza el proceso de adición de nuevas decoraciones al juego, que se desbloquearán de forma aleatoria al finalizar un nivel del juego. El *script* de control de las decoraciones incluye la creación de un lugar de cámara para visualizar el objeto desbloqueado. En caso de que la visualización de este objeto al desbloquearse no sea perfecta, puede modificarse la posición o rotación del objeto en la escena.

ANEXO V: Presupuesto

A continuación, se valoran los recursos materiales, personales y de licencias de software empleados para calcular la inversión para realizar el proyecto de forma estimada.

Se ha necesitado un ordenador con el sistema operativo Windows 10, un mando compatible (Xbox) y un equipo de grabación básico conformado por un micrófono, una interfaz de audio y auriculares.

Elordenador utilizado es un Asus Strix GL502VMK que consta de:

- Sistema Operativo: Microsoft Windows 10 Home 64 Bit
- Procesador: Intel Core i7-7700HQ 4 x 2.8 - 3.8 GHz, Kaby Lake
- Adaptador gráfico: NVIDIA GeForce GTX 1060 Mobile
- Memoria RAM: 12 GB
- Disco duro: 128 GB SSD + 1 TB HDD
- Conexiones: 3 USB 3.0, 1 Thunderbolt, 1 HDMI, 1 DisplayPort
- Batería: 64 Wh, 4100 mAh

Equipo de grabación:

- Tarjeta de sonido: Focusrite Scarlett 2i2 2nd Gen
- Micrófono: Shure sm58
- Auriculares: AKG K-240 Studio
- Mando: Diswoe Xbox 360

Además, se ha hecho uso del siguiente software:

- Unity: Motor de juego con licencia gratuita, utilizado para la implementación de los elementos del juego y su lógica interna
- Visual Studio 2019: Es un entorno de desarrollo integrado (IDE) completo que puede usar para escribir, editar, depurar y compilar el código y, luego, implementar la aplicación.
- GIMP 2.10: programa de edición de imágenes digitales. Utilizado para la creación de elementos de la interfaz de usuario y edición de texturas.
- Blender 3.5: programa de modelado y animación. Utilizado para la creación y animación de modelos 3D, así como la modificación de materiales externos para adecuarlos a las necesidades del proyecto.
- Reaper 6.14: estación de trabajo de audio digital. Utilizado para la grabación y edición de los elementos sonoros presentes en el juego.

El modelo 3D de los guacamayos es un modelo de pago adquirido en Turbosquid. Se trata del modelo “Blue Bird” de Alien Interactive [1].

El desarrollo de este proyecto se ha prolongado a lo largo de 6 meses y se han dedicado un total de 323 horas. En el momento de realización de este proyecto, las estimaciones del salario bruto de un ingeniero junior indican una media de 25.000,00€ anuales en el mercado laboral actual para una jornada laboral de 40 horas semanales [2]. Por lo tanto, teniendo en cuenta las horas empleadas en este proyecto se obtiene la suma de 3.882,00€ como coste personal.

Con todos los datos recogidos, se crea un presupuesto total para el desarrollo del proyecto (Tabla 1).

Anexo V. Tabla 1: Presupuesto para el desarrollo del proyecto.

DESCRIPCIÓN		UNIDADES	COSTE
COSTE PERSONAL	Ingeniero junior	1	3.882,00€
		PARCIAL	3.882,00€
COSTE MATERIAL	Asus Strix GL502VMK	1	800,00€
	Diswoe Xbox 360	1	32,00€
	Focusrite Scarlett 2i2	1	144,00€
	AKG K-240 Studio	1	72,00€
	Shure sm58	1	109,00€
		PARCIAL	1.157,00€
COSTE DE LICENCIAS	Unity	1	0,00€
	Visual Studio	1	0,00€
	GIMP	1	0,00€
	Blender	1	0,00€
	Reaper	1	225,00€
	Modelo “Blue Bird” Alien Interactive	1	10,00€
		PARCIAL	235,00€
		TOTAL	5.274,00€

Referencias

[1] Alien Interactive, «Turbosquid,» [En línea]. Available: https://www.turbosquid.com/es/3d-models/3d-bird-parrot-blue-ar-1301496. [Último acceso: 11 julio 2023].

[2] « Salario medio para Ingeniero Junior en España,» [En línea]. Available: https://es.talent.com/salary?job=ingeniero%2Bjunior. [Último acceso: 11 julio 2023].