PROYECTO FIN DE GRADO

TÍTULO: Ampliación de un videojuego terapéutico para la mejora del procesamiento auditivo: desarrollo de un módulo de secuencias melódicas

AUTOR: Mario Ortega García

TITULACIÓN: Grado en Ingeniería de Sonido e Imagen

TUTORA: Martina Eckert

DEPARTAMENTO: Ingeniería Audiovisual y Comunicaciones

VºBº TUTORA

Miembros del Tribunal Calificador:

PRESIDENTA: Inmaculada Álvarez de Mon Rego

TUTORA: Martina Eckert

SECRETARIO: Enrique Rendón Angulo

Fecha de lectura: 26 de julio de 2023

Calificación:

El Secretario,

Agradecimientos

A mis padres, a mis amigos y a Inés, por creer en mí y ser un apoyo indispensable durante todos estos años.

Resumen

El objetivo de este proyecto fin de grado es la ampliación del videojuego terapéutico “El Planeta Sonoa” mediante la creación de un nuevo módulo. “El Planeta Sonoa” está dirigido a menores de entre 5 y 14 años diagnosticados con Desorden del Procesamiento Auditivo Central (DPAC). El DPAC se define como un trastorno que dificulta el procesamiento de la información auditiva, especialmente en ambientes ruidosos. La misión de “El Planeta Sonoa” es amenizar las sesiones de terapia de los niños que lo padecen mediante una gamificación de los ejercicios de entrenamiento auditivo. Está formado por distintos módulos, cada uno de los cuáles se centra en reforzar una o más de las habilidades que resultan dificultosas para las personas con DPAC. Además, cada uno de estos juegos transcurre en un continente diferente.

El módulo que concierne a este proyecto se centra en la discriminación y memoria auditivas. La discriminación auditiva consiste en diferenciar sonidos de distinta duración, frecuencia o nivel, mientras que la memoria auditiva consiste en retener la información percibida. Para entrenar estas capacidades, se ha desarrollado un juego que consiste en memorizar y repetir secuencias de notas musicales de distinta duración y tono. Ha sido desarrollado en el motor y editor de videojuegos Unity. Para la creación y edición de modelos 3D se ha usado el software Blender, para la edición de imágenes, GIMP, y para la edición de audio, Audacity, Reaper y Ableton Live.

El juego consta de una dificultad ajustable mediante distintos parámetros, como el número de notas de los patrones que se solicitan, la velocidad de reproducción de estos, o el volumen de los ruidos de distracción, entre otros. Estas variables pueden ajustarse tanto desde el menú interno del juego, como desde una interfaz web con base de datos asociada al juego. La web está dirigida a los terapeutas que supervisan las sesiones de juego de los pacientes. Desde ella, se pueden establecer los ajustes de juego individualmente, según las necesidades de cada usuario. De la misma forma, permite revisar los resultados de las sesiones de juego anteriores para evaluar el progreso del paciente.

En cuanto al apartado estético, se ha procurado que el juego resulte vistoso y animado para el público objetivo. Transcurre en el continente ficticio “Kobarag”, una región desértica en la que destacan su oasis y su mercado. Dos de las cobras del mercado están tristes y será la misión del jugador animarlas tocando sus melodías favoritas con un *pungi.*

Abstract

The objective of this bachelor's degree final project is the expansion of the therapeutic videogame "El Planeta Sonoa" by creating a new module. "El Planeta Sonoa" is geared towards minors between the ages of 5 and 14 who have been diagnosed with Central Auditory Processing Disorder (CAPD). CAPD is defined as a disorder which hinders the processing of auditory information, especially in noisy environments. The aim of the game is to increase the enjoyability of therapy sessions through the gamification of auditory training exercises. It is composed of several modules, each of which centres around reinforcing one or more of the abilities which tend to be difficult for people with CAPD. Furthermore, each of these games takes place in a different continent.

The module this project is centred around focuses on auditory discrimination and memory. Auditory discrimination allows people to differentiate sounds of different lengths, frequencies, or level, while auditory memory is a about retaining the perceived information. To train these abilities, a game that revolves around memorizing and repeating sequences of musical notes of different duration and tone has been developed. In order to do so, the Unity videogame engine and editor was used. Blender was used to create and edit 3D models, GIMP to edit images and Audacity, Reaper and Ableton Live to edit audio.

The game has an adjustable difficulty through different parameters, like the number of notes of the patterns played, playback speed or the volume of distraction noises, among others. These variables can be adjusted both through the internal menu of the game and through the associated website and database. This website is aimed at the therapists that supervise the patient's gaming sessions. Through it, game settings can be established individually, according to each user's needs. Likewise, it allows the therapists to review the results of past gaming sessions to evaluate the patient's progress.

From the aesthetical point of view, an effort was made to make the game feel eye-catching and lively. It takes places in the fictional continent of "Kobarag", a desertic region known for its oasis and market. Two of the market's cobras are sad and it is the players mission to lift their spirits by playing their favourite melodies with their *pungi*.

Acrónimos

DPAC: Desorden del Procesamiento Auditivo Central

SNAC: Sistema Nervioso Auditivo Central

ETSIST: Escuela Técnica Superior de Ingeniería y Sistemas de Telecomunicación

GAMMA: Grupo de investigación de Aplicaciones MultiMedia y Acústica

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

NPC: *Non-Playable Character*

MIDI: *Musical Instrument Digital Interface*

DAW: *Digital Audio Workstation*

UI: *User Interface*

GDD: *Game Design Document*

Índice

Índice de figuras

Índice de tablas

1. Introducción

A lo largo de las últimas dos décadas la industria de los videojuegos ha sufrido una metamorfosis debido a la democratización del proceso de desarrollo [1]. Esto se debe en gran medida a la aparición de motores gráficos gratuitos como Unity, Unreal Engine o Godot, entre muchos otros, así como la ingente cantidad de recursos e información accesible en internet. Esto ha permitido que se puedan crear videojuegos sin contar necesariamente con un extenso grupo de desarrolladores o un gran presupuesto. En consecuencia, lo que históricamente se ha considerado un medio enfocado en ofrecer ocio a las masas, ha abierto sus puertas a experiencias artísticas, narrativas o didácticas que no hubieran sido posibles de otra forma. Entre estos nuevos subgéneros se encuentran los videojuegos serios o terapéuticos.

Los videojuegos serios son una herramienta útil, ya que ayudan a los pacientes a afrontar la terapia desde un ángulo diferente, a menudo más divertido y motivador. Sin embargo, hoy en día siguen siendo un concepto relativamente joven, y, por tanto, escaso. Esto resulta especialmente cierto en el caso del Desorden del Procesamiento Auditivo Central (DPAC), el trastorno entorno al cual se centra este proyecto. Los terapeutas que tratan este desorden afirman que apenas existen recursos en castellano especializados que puedan usar en las sesiones de terapia. De esta necesidad nace “El Planeta Sonoa”, un videojuego terapéutico dirigido a menores de 5 a 14 años dispuesto a mejorar las capacidades de procesamiento auditivo de sus usuarios.

“El Planeta Sonoa” tiene una estructura modular, y cada una de sus partes se centra en reforzar una o más habilidades que resultan difíciles para las personas con DPAC. El objetivo principal de este proyecto es el desarrollo de un nuevo módulo para este videojuego que ayude a mejorar la discriminación y memoria auditivas de los usuarios. Este nuevo módulo debe respetar la temática, estética y estructura de niveles marcadas por los juegos desarrollados anteriormente, facilitando así su integración. Para su elaboración debe usarse el motor de videojuegos Unity (versión 2021.3) y el lenguaje de programación C#.

En este documento se explican en primer lugar los síntomas y repercusiones del Desorden del Procesamiento Auditivo Central, y cómo el “El Planeta Sonoa” pretende paliarlos. Tras esto, se listan los objetivos específicos del nuevo módulo desarrollado, y se realiza una descripción detallada de la solución propuesta. En esta descripción se comienza por los aspectos más básicos como son la creación del entorno y personajes, o la estructura del juego. Posteriormente, se dan detalles sobre elementos más específicos como la lógica interna, los controles, el diseño de sonido o los menús. Finalmente, se analizará el impacto socioambiental del proyecto y se aportarán las conclusiones generales a las que se han llegado, así como posibles líneas de trabajo futuro.

2. Antecedentes y marco tecnológico

El videojuego que se ha desarrollado es una ampliación de "El Planeta Sonoa", un proyecto del Grupo de investigación de Aplicaciones MultiMedia y Acústica (GAMMA), que a su vez forma parte del Centro de Investigación en Tecnologías Software y Sistemas Multimedia para la Sostenibilidad (CITSEM). Una de las principales líneas de investigación de GAMMA es la gamificación de las sesiones de terapia de menores con diversidad funcional mediante el desarrollo de videojuegos serios. Además de "El Planeta Sonoa", este grupo de investigación comenzó en 2018 el desarrollo de "Phiby's Adventures 3D", un videojuego terapéutico enfocado a niños con movilidad reducida [2].

En cuanto a videojuegos serios centrados en el procesamiento auditivo, se han encontrado dos ejemplos. En primer lugar, *Sound Storm*, un juego que pretende ayudar a niños de entre 6 y 12 años diagnosticados con el Desorden del Procesamiento Espacial [3]. Este trastorno dificulta la ubicación de sonidos en el espacio y, consecuentemente, la separación entre la información auditiva relevante e irrelevante. Por otro lado, *Fast Forword* es un *software* enfocado a niños con dislexia, necesidades especiales u otros factores que dificultan el aprendizaje [4]. Emplea juegos centrados en el entrenamiento auditivo para mejorar las capacidades lingüísticas de estudiantes de primaria y secundaria. Al igual que "El Planeta Sonoa", ambos ejemplos citados se basan en la neuroplasticidad y el entrenamiento auditivo para reforzar las facultades de procesamiento auditivo de los usuarios.

2.1 Desorden del Procesamiento Auditivo Central

Investigaciones acerca de la neuroplasticidad han demostrado que el entrenamiento del Sistema Nervioso Auditivo Central (SNAC) es comparable al de un músculo. Esto es, de la misma forma que la ausencia de estímulos puede causar alteraciones negativas tanto estructurales como funcionales en el SNAC, la exposición repetida a determinados estímulos puede resultar en una mejora funcional [5] [6]. Mediante la gamificación de este entrenamiento, se pretende ofrecer a los pacientes una forma amena de realizar las actividades terapéuticas repetitivas que eventualmente darían lugar a un alivio de sus síntomas.

El Desorden de Procesamiento Auditivo Central se define como un déficit en el procesamiento de la información de carácter auditivo. Una de las conductas más habituales en pacientes con DPAC es la dificultad para comprender y mantener la atención en ambientes ruidosos. Existen también una serie de retos relacionados con el lenguaje, como la lectura, escritura, tomar notas o aprender nuevo vocabulario. Otros problemas más relevantes para este módulo de secuencias melódicas son el déficit de memoria auditiva y la dificultad en el procesamiento de señales no verbales, como la música.

La estrecha relación entre el procesamiento de la información auditiva y la adquisición del lenguaje oral puede dificultar en gran medida el aprendizaje de niños con DPAC. La tabla 1, extraída del artículo de Óscar Cañete presenta las habilidades auditivas más importantes en el proceso de aprendizaje [7].

Tabla 1. Habilidades auditivas importantes en el proceso de aprendizaje.

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

Cada uno de los módulos de "El Planeta Sonoa" pretende proporcionar a los usuarios ejercicios para entrenar una o más habilidades presentes en la tabla. Este módulo de secuencias melódicas se centra en la discriminación de sonidos de distinta duración y frecuencia, así como la memoria y la separación auditivas en ambientes ruidosos.

2.2 El Planeta Sonoa

Una vez se han presentado los síntomas y dificultades a los que se enfrentan las personas diagnosticadas con DPAC, se va a detallar en qué consiste "El Planeta Sonoa". Este proyecto, iniciado en 2018 y dirigido por la profesora Martina Eckert, recoge distintos juegos terapéuticos desarrollados por alumnos de la Escuela Técnica Superior de Ingeniería y Sistemas de Telecomunicación (ETSIST). Cada uno de los juegos está enfocado a reforzar una capacidad que suele resultar dificultosa para las personas diagnosticadas con DPAC. A su vez, cada ejercicio está ambientado en un continente ficticio diferente, en los que el jugador conocerá a un elenco de animales parlantes a los que debe ayudar mediante la realización de ejercicios auditivos. Los módulos que conforman el proyecto son:

- Polaris. Este módulo está inspirado en la Antártida y fue desarrollado por Luna Sampedro Jiménez [8] y ampliado por Esther García Medina en prácticas [9]. El juego pretende

trabajar la memoria auditiva secuencial y la discriminación de señales no verbales en ambientes de ruido.

- Selvalia. Con una selva tropical como escenario, este videojuego desarrollado por María Cristina Ojeda Egocheaga sirve como entrenamiento para la discriminación de fonemas acústicamente similares [10].
- Cavernia. Desarrollado por Bogdán Morar, este juego transcurre en una cueva en la que se relatan historias, con la intención de que el jugador mejore su atención y asociación auditiva [11].
- Tropikus. Ambientado en una isla tropical, este módulo permite entrenar la escucha dicótica (la percepción simultánea de dos estímulos diferentes, uno en cada oído). Ha sido desarrollado por Pablo Lucas Olivares [12].
- Pandalia. Inspirado por la cultura asiática, este juego de realidad virtual ha sido desarrollado por Moheng Li. Se emplean sonidos 3D para trabajar la ubicación de fuentes sonoras en el espacio [13].

Todos estos continentes han sido integrados en un mismo proyecto de Unity tras finalizar su desarrollo, constituyendo así "El Planeta Sonoa". El menú inicial del juego presenta un globo terráqueo en 3D que permite viajar entre los distintos continentes, es decir, seleccionar la prueba de entrenamiento auditivo que se va a realizar (figura 1).

Figura 1. Menú de selección de continente.

Existe además una interfaz web para el proyecto "El Planeta Sonoa". Esta web está pensada para ser usada únicamente por los terapeutas y audiólogos mediante su nombre de usuario y contraseña privados. A su vez, cada paciente cuenta con su propio nombre de usuario y contraseña. De esta manera, los terapeutas podrán acceder a los perfiles de cada paciente y configurar los ajustes de cada juego de manera individual, además de consultar los resultados de sus sesiones de juego. En caso de que los jugadores no dispongan de conexión a internet o no estén registrados en la base de datos, se podrá jugar también modo local. Tanto la integración de los módulos como la interfaz web han sido desarrolladas por Miguel Jiménez Arévalo.

A los cinco módulos mencionados anteriormente se suma Kobarag, el continente en el que transcurre el juego que concierne a este Proyecto Fin de Grado. Este módulo, ambientado en un desierto, se centra en la discriminación de notas musicales de distinta duración y frecuencia, así

como en la memoria auditiva. Durante el desarrollo del proyecto se ha contado con la colaboración de terapeutas del Centro de Audiología y Logopedia Áurea TAV Madrid, a las que se ha consultado para asegurar la eficacia de los ejercicios presentes en el videojuego.

3. Objetivos del proyecto

Tras haber explicado el concepto de "El Planeta Sonoa" y así como los síntomas del DPAC, se plantean los objetivos que pretende cumplir este módulo en concreto.

1. Este módulo debe proporcionar ejercicios que permitan a los usuarios entrenar su capacidad de discriminación y memoria auditivas. Para ello, se deben generar secuencias de notas de distinta duración y tono, que el jugador debe memorizar y repetir.
2. El juego debe ser divertido y animado para el público objetivo, niños y niñas de 5 a 14 años. Deben establecerse distintos objetivos y recompensas que motiven al usuario a seguir jugando y mejorando. La principal ventaja de un videojuego terapéutico frente a tratamientos más convencionales es que puede resultar más atractivo y ameno, y, por tanto, ha de aprovecharse este aspecto.
3. Deben existir distintos parámetros configurables que permitan ajustar la dificultad del juego a las necesidades de cada paciente. Cada usuario puede tener capacidades y dificultades radicalmente distintas, por lo que la experiencia de juego debería ser lo más personalizable posible. De esta forma se permite además aumentar la dificultad de las pruebas gradualmente si el terapeuta observa progreso.
4. El videojuego debe tener controles intuitivos y poder controlarse tanto con teclado y ratón como con mando. Al estar dirigido a niños, resulta esencial que se dote al proyecto de todas las facilidades posibles a la hora de moverse e interactuar con un entorno 3D.
5. Se debe establecer conectividad entre la página web de "El Planeta Sonoa" y este módulo, de manera que un terapeuta pueda configurar los ajustes de un paciente desde cualquier dispositivo inteligente. A su vez, los resultados de cada sesión de juego deberán quedar registrados en la web para su posterior consulta por parte del terapeuta.

4. Descripción de las solución propuesta

En esta sección se detalla el desarrollo del módulo “Kobarag” de “El Planeta Sonoa”. Se abarca el proceso de elaboración completo, desde la concepción de las ideas más básicas a los últimos detalles, justificando cada decisión de diseño. Se comienza explicando los apartados más fundamentales, como son el concepto general y la estructura del juego. Posteriormente, se indaga en aspectos más específicos como el funcionamiento de determinados scripts, el diseño de audio o la conectividad con la página web.

4.1 Entorno y personajes

La idea fundamental de “El Planeta Sonoa” es que el jugador viaja alrededor de un mundo poblado por animales parlantes, visitando distintos continentes y ecosistemas. A la hora de elegir la ambientación en la que transcurriría el módulo de patrones melódicos, se partió de la idea del encantamiento de serpientes, práctica típica de la India. Tocar las notas correctas con un instrumento musical para hacer bailar a una cobra resultaba ser una premisa divertida y que encajaba bien tanto con la temática del juego, como con las mecánicas que se querían implementar.

El encantamiento de serpientes se suele realizar mediante un *pungi,* instrumento de viento tradicional indio. La versión más básica del mismo consiste en una calabaza seca a la que se le conectan dos cañas de bambú usando cera de abeja [14].

Este espectáculo suele ser realizado por artistas callejeros en mercados y festivales indios. Siendo además la India el país en el que se encuentra el desierto del Thar, uno de los más grandes del mundo, se acabaron combinando estas ideas para diseñar Kobarag, el desierto ficticio en el que tiene lugar el juego. El nombre proviene del hindi *kobara* (कोबरा), que siginifica “cobra” y *raag* (राग), que significa “melodía”. En la figura 2 se muestra el primer plano que se planteó para Kobarag en la fase conceptual del juego.

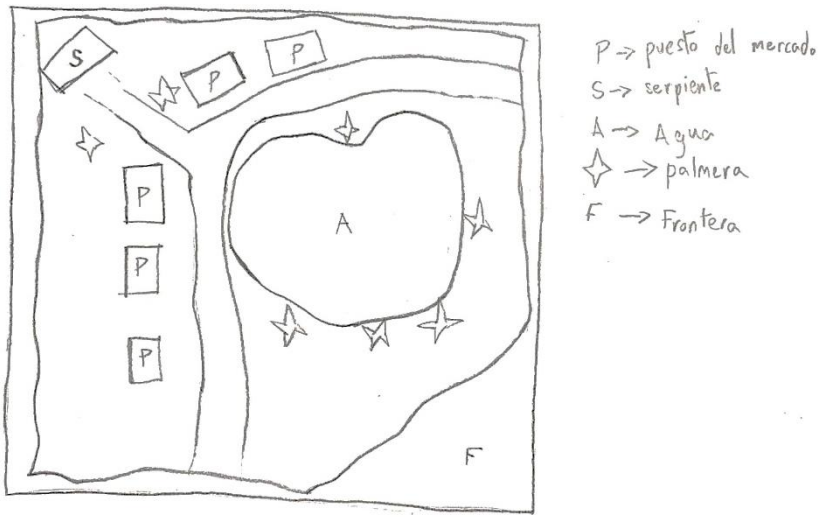

Figura 2. Primer plano del desierto en la fase conceptual (6 de febrero de 2023).

4.1.1 Diseño de personajes

Kobarag es una región desértica, pero de gran importancia comercial debido al mercado que se formó alrededor de su oasis. Este mercado está regentado por distintos animales, pero principalmente cobras. Dos de ellas están desanimadas ya que llevan mucho tiempo sin escuchar sus melodías favoritas, y será la misión del jugador hacerlas bailar tocando las melodías correctas. A continuación, se van a comentar cada uno de los personajes que habitan el mercado:

- Dora: esta cobra verde pide al jugador que repita sus melodías formadas por notas de distinta duración.
- Antón: una cobra morada que pide al jugador que repita sus melodías constituidas por notas de distinto tono.

En consecuencia, la mayor parte del tiempo de juego tendrá lugar frente a estos dos NPCs (*Non-Playable Characters*), ya que el objetivo principal consiste en repetir sus melodías. Interactuar con Dora y Antón es el ejercicio terapéutico propiamente dicho. Sin embargo, existen otros personajes secundarios cuya función comunicar información al usuario o recompensarle por su esfuerzo.

- Camilo: se trata de un camello cuya función principal es la de guiar al jugador. Aparece en la cinemática inicial para presentar el contexto del juego y dar algunas indicaciones sobre lo que se debe hacer. Por último, otorga el *pungi* diamante al jugador como recompensa final.
- Patel: esta cobra tendera con sombrero no está interesada en la música. En cambio, ayuda al jugador vendiéndole el *pungi* dorado, y dándole pistas sobre cómo desbloquear el *pungi cactus*. Por tanto, su función principal es hacer saber al jugador que existen diversos instrumentos en el juego que se pueden obtener a cambio de gemas si se explora el entorno y se habla con los personajes.

Los modelos 3D de autoría propia en este videojuego son las cobras principales (verde y morada), que son el mismo modelo con diferente textura; el buitre, el *pungi*, el poste de madera y la cuerda. Estos dos últimos son mucho más básicos ya que su función se limita a sostener las lámparas que cuelgan sobre el mercado, por lo que no se indagará en ellos. Todos los modelos 3D mencionados se han esculpido usando el *software* Blender, versión 3.4.

El primer modelo que se realizó fue el de las cobras principales, Dora y Antón. Desde un primer momento se buscó un estilo *low poly* (de pocos polígonos, lo cual favorece al rendimiento del juego) y *cartoon.* Otro requisito indispensable era que la serpiente tuviera una mandíbula que pudiera abrirse y cerrarse, ya que debía hablar y cantar. Inicialmente la cobra se iba a elevar con cada nota acertada por el jugador. Por este motivo, se elaboró el modelo de manera que fuera sumamente largo, para que el jugador nunca viera el extremo inferior de la cobra. Si se diera ese caso, daría la impresión de que está levitando y se rompería la ilusión de la cobra está contenida en la cesta. Finalmente, se decidió que sólo se alzase al terminar una ronda, ya que el planteamiento incial parecía excesivo.

Para modelar al personaje se siguió el procedimiento mostrado por Blender Default Cube en su tutorial [15] . Se comienza formando una especie de “columna vertebral” mediante una concatenación de vértices a los que posteriormente se aplica el modificador *Skin* para darles volumen. Otros modificadores que se emplean son *Mirror*, para que el resultado quede simétrico

y *Subdivision Surface,* que suaviza el aspecto del modelo subdividiendo los polígonos que lo forman. Las texturas se pintaron manualmente usando la ventana *Texture Paint* de Blender. Mediante máscaras, se separaron cada una de las secciones de la cobra que requieren colores diferentes (boca, colmillos, ojos, cuerpo exterior y cuerpo interior) para poder colorearlas individualmente sin manchar otras zonas. En la figura 3 se puede apreciar el resultado final.

Figura 3. Modelo 3D de las cobras principales. De izquierda a derecha: la cobra morada de perfil, la cobra verde de frente y extensión total del modelo.

El segundo modelo más complejo de autoría propia es el buitre. Este animal no es un personaje, sino que su función es desconcentrar al jugador mediante sus graznidos. Para modelarlo se siguió también un tutorial de Blender Default Cube [16]. En este caso, se importó a Blender una fotografía de un buitre y se colocó en uno de los ejes de simetría de un cubo. Aplicando el modificador *Mirror* para mantener la simetría, se deformó esta figura inicial para hacerla coincidir con la imagen de referencia, aplicando diversas herramientas en el proceso. Finalmente se pintó la textura de manera análoga al caso de la cobra. El modelo final del buitre se muestra en la figura 4.

Figura 4. Modelo 3D del buitre. A la izquierda, el perfil; a la derecha, la vista frontal.

El último modelo que se va a comentar de los mencionados es el *pungi*, aunque comparativamente es más sencillo que los anteriores. Está formado por una esfera y dos cilindros. La esfera fue deformada para darle forma de calabaza con un orificio por el que soplar. Por su parte, los cilindros. La parte más complicada fue realizar los orificios equidistantes en uno de los cilindros, donde se sitúan los dedos. Para ello, se empleó el modificador *Array* para generar una serie de pequeños cilindros equidistantes que atravesaban el cilindro principal. Posteriormente, se les aplicó el modificador *Boolean*, que sustituye estos cilindros por los orificios. La textura de madera fue pintada manualmente. El modelo generado se muestra en la figura 5.

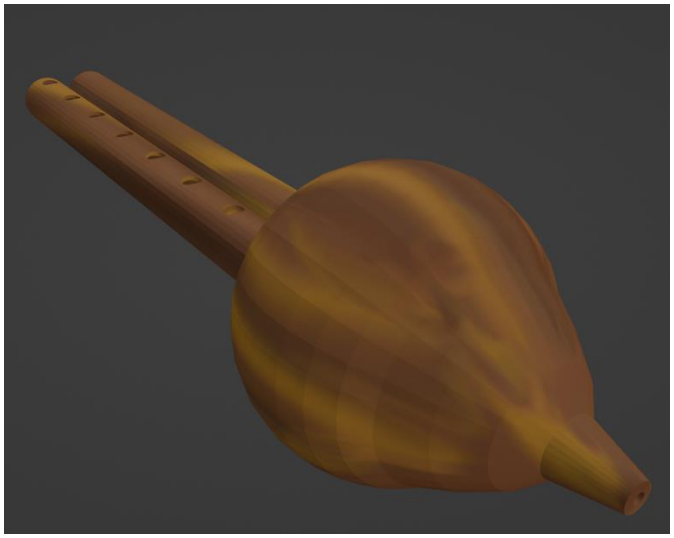

Figura 5. Modelo 3D del pungi de madera.

El resto de modelos se obtuvieron de diversas plataformas especializadas en compartir objetos 3D, revisando siempre que tuvieran una licencia apropiada por parte del autor. Sin embargo, algunos de estos modelos fueron editados, ya sea por razones estéticas o prácticas. Por ejemplo, el modelo del camello fue suavizado utilizando la herramienta *Subdivision Surface* de Blender, y se le modeló una boca para darle expresividad y facilitar que el jugador ubicase al orador. A la cobra tendera se le eliminó manualmente la pistola que originalmente tenía en su cartuchera, para hacerla menos violenta y más apropiada para un público infantil.

Tras elaborar y obtener los modelos 3D indicados, el siguiente paso es animarlos. Pero para ello es necesario en primer lugar crear una armadura. Las armaduras están formadas por huesos, los cuales se asocian a distintas partes de los modelos 3D. De esta forma, cuando un determinado hueso se rote o se desplace, también lo hará la malla del modelo a su alrededor. En la figura 6 se exponen las armaduras creadas para dos de los personajes del juego. Se puede apreciar que algunos huesos son de color verde. Esto indica que se les ha aplicado el modificador *Child Of.* Este modificador hace que dicho hueso dependa de otro hueso (que será su "padre") aunque no estén físicamente conectados. Por ejemplo, en el caso de la cobra, su mandíbula está inconexa del resto de huesos. Pero dado que la mandíbula debe moverse con el resto de la cabeza, se ha aplicado este modificador.

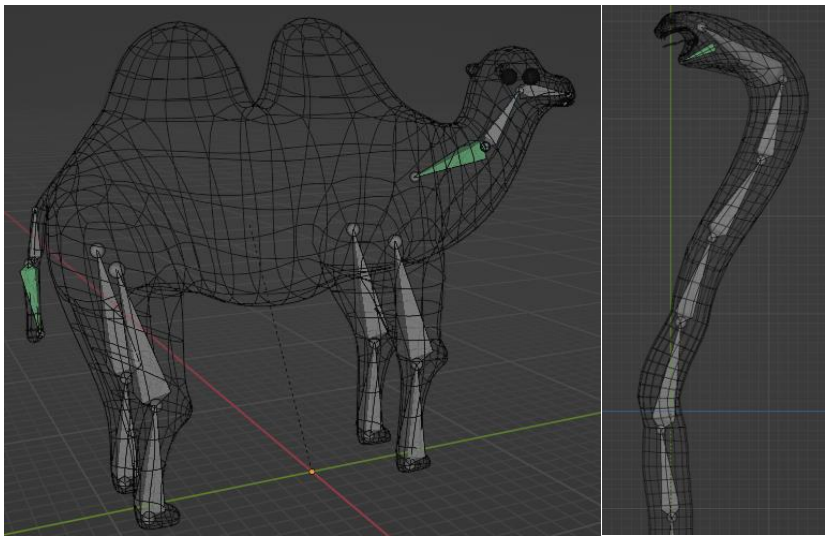

Figura 6. Armaduras del camello (izquierda) y la cobra (derecha).

Tras elaborar armaduras para las cobras principales, el camello, la cobra tendera y el buitre, finalmente pueden animarse. Para ello se ha empleado el *Action Editor* de Blender. Esta ventana es una línea temporal con un canal para cada uno de los huesos del modelo. Si un hueso se rota o se desplaza en un fotograma determinado de la línea temporal, se crea un *keyframe* (un fotograma clave). Estos se ven representados como puntos en la figura 7. Esencialmente, la animación 3D consiste en usar los huesos del personaje para colocarlo en una serie de poses clave, y la transición entre una pose y la siguiente la calcula el programa mediante interpolación. La ventaja del *Action Editor* es que, como su nombre indica, está pensado para animaciones de acciones concretas. Por tanto, para cada modelo deberán planearse qué acciones llevará a cabo durante el desarrollo del juego.

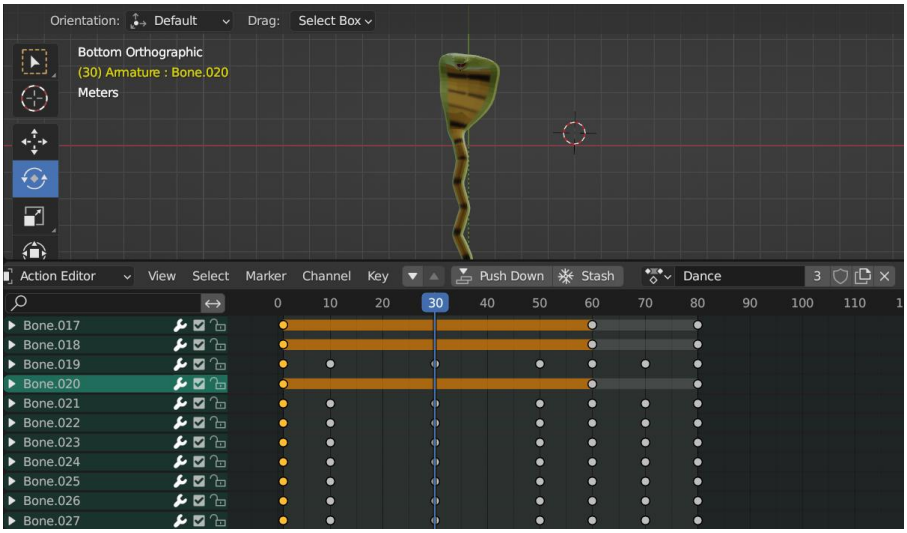

Figura 7. Action Editor de Blender.

Prácticamente todas las animaciones que aparecen en el juego son de autoría propia. La única excepción es la de apertura y cierre del cofre en el que se encuentra el *pungi* cactus [17]. A continuación, se listan todas las animaciones realizadas:

- El modelo de las cobras principales consta de siete animaciones diferentes: hablar, bailar, cantar, dos tipos de celebración, confusión y *idle* (la animación que se reproduce cuando el personaje está "quieto").
- El camello tiene animación *idle*, de hablar y de caminar.
- La cobra tendera tiene animaciones *idle* y de hablar.
- Para el buitre se realizaron dos animaciones: *idle* y una de cacareo que finalmente no se utilizó en el juego.

Para cada modelo 3D animado, es necesario crear un *Animation Controller* en Unity. El *Animation Controller* permite especificar cómo y cuándo se realizan transiciones entre distintas animaciones. En la figura 8 se puede observar el controlador de animaciones más complejo de todos, asignado a Dora y Antón. Inicialmente se situó *idle* en el centro, por ser la animación neutral, y todas las animaciones debían partir de ella y regresar a ella tras finalizar. Posteriormente, se añadieron transiciones entre las acciones de bailar, hablar y cantar, ya que estas ocurren de manera muy seguida durante las pruebas de memoria auditiva.

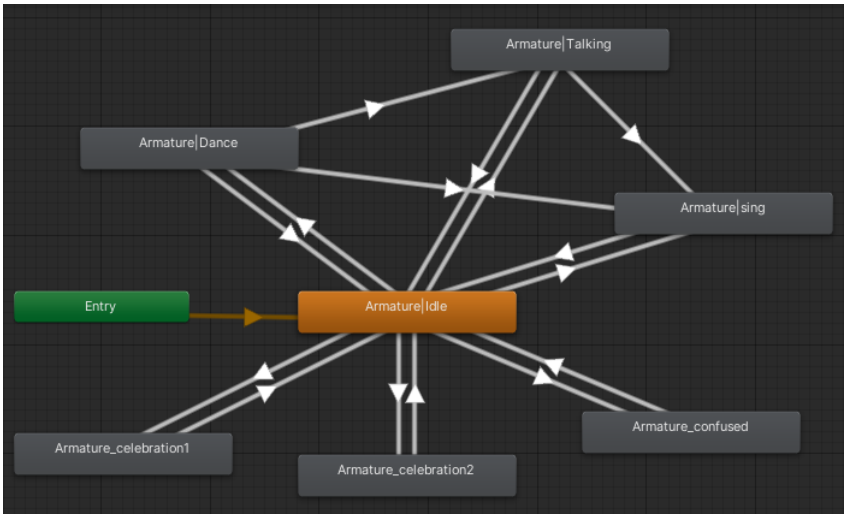

Figura 8. Animation Controller de las cobras principales.

Por último, siendo este un videojuego basado en la audición parecía indispensable que los personajes se comunicasen mediante voz y no mediante texto. Por este motivo, todos los personajes del juego han sido doblados. La voz de Dora es Blanca Gómez-Soro Jiménez y la voz de Antón es Juan Sevilla Sierra. En cuanto a los personajes secundarios, Patel fue doblado por Raúl J. Sánchez y Camilo por Mario Ortega García. Las grabaciones fueron realizadas mediante un micrófono Rode NT1-A y una interfaz de audio Steinberg UR22 en el *software* Ableton Live 10. El guion completo del videojuego se puede leer en el apartado Anexo III: Guion.

4.1.2 Diseño del entorno

Debido a que el objetivo del juego es bastante concreto y no requiere un mapa de gran tamaño, el terreno sobre el que se ha trabajado en este proyecto es de una extensión moderada. Concretamente, tiene unas medidas de 120 x 120 en cuanto a superficie, y un rango de altura de 700, ya que consta de dunas de distintos tamaños y una cavidad en la que se sitúa el oasis. Para modelar el relieve del terreno se usó el editor de Unity. Marcando la opción *Raise or Lower Terrain* se pueden usar una variedad de pinceles que modifican la altura de la malla a su paso.

En cuanto a la textura del desierto, el terreno consta de tres capas. En primer lugar, se usó un color básico incluido en el paquete *Free Low Poly Desert Pack* de la *Unity Asset Store* [18]. Posteriormente, se aplicó una textura de arena obtenida de *SketchUpTextureClub* [19] y otra de terreno árido obtenida de *FreePik* [20]. Ambas texturas son *seamless*, lo que quiere decir que se pueden colocar en forma de mosaico sin que se perciba división entre una imagen y la siguiente. Se ajustaron las opacidades de cada capa del terreno de manera que todas fueran visibles, pero sin resaltar demasiado. La figura 9 muestra una vista cenital del desierto.

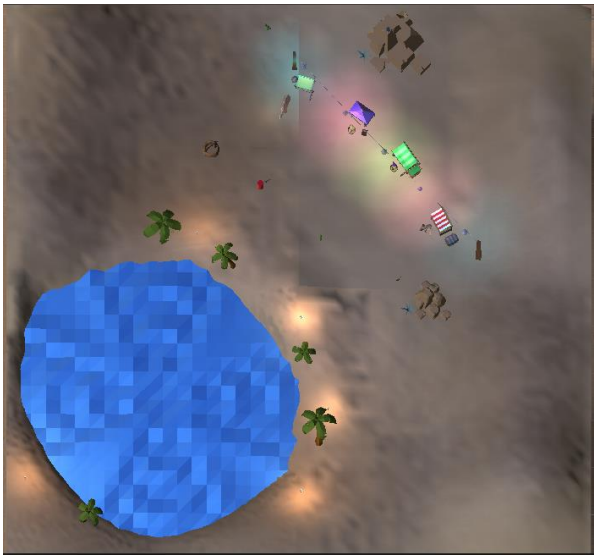

Figura 9. Vista cenital del entorno diseñado en Unity.

En cuanto al estanque del oasis, se empleó el paquete *Low Poly Water* de Ebru Dogan, disponible en la *Unity Asset Store* [21]. Este paquete permite generar planos de mallas a los que se les puede aplicar un efecto de olas mediante un script, pudiendo configurar el nivel de detalle, frecuencia y tamaño de las ondulaciones.

El mercado, por su parte, está constituido por tres puestos creados por Makalina (compartidos en *CGTrader)* [22], y uno sacado del paquete *Environment Lite,* de la *Unity Asset Store* [23]. La obra de Makalina incluía además una variedad de modelos 3D que se usaron para decorar los puestos y darles vida, como carteles, sacos, verduras o un barril. La figura 10 muestra el aspecto final del mercado. Junto al mercado se encuentra un pozo, obtenido del paquete *Free Low Poly Desert* mencionado anteriormente, pero editado en Blender para suavizarlo y aumentar el número de polígonos, ya que originalmente era extremadamente bajo.

Figura 10. Mercado de Kobarag.

El resto de la decoración está constituida por la flora (palmeras y cactus principalmente), antorchas y montículos de roca. En la figura 11 se aprecia el aspecto del oasis, rodeado por palmeras y antorchas. Es necesario aclarar que las plantas escogidas para decorar la escena no coinciden necesariamente con las autóctonas del desierto del Thar o la India en general. No obstante, al ser este un desierto ficticio, y siendo las palmeras y cactus la vegetación más comúnmente asociada a los desiertos, se decidió tomar esta licencia creativa.

Figura 11. Oasis de Kobarag.

Tal y como se ha mencionado, existen antorchas, además de varias lámparas, iluminando las escena, ya que esta transcurre durante la noche. Se tomó esta decisión por varios motivos. En primer lugar, visualmente resulta en un mayor contraste entre el cielo azul oscuro y el suelo de arena, de un marrón amarillento cálido. A su vez, se suavizan los colores de todos los elementos en pantalla, en comparación a cómo se veían cuando la *Directional Light* (la luz principal de la escena) tenía mayor intensidad. Por último, le da una ambientación distinta al resto de módulos del proyecto y permite jugar con las luces.

Las lámparas de estilo árabe desprenden luz de distintos colores, de acuerdo con los de sus cristales. Esto queda muy vistoso, en contraste con el desierto, que es un ecosistema árido por definición. Las antorchas por su parte desprenden una luz cálida y agradable. Sin embargo, los personajes del juego seguían quedando algo oscuros y difíciles de apreciar, por lo que se les colocaron luces de baja intensidad en puntos estratégicos. Todas las luces existentes en el juego (excluyendo la *Directional Light* principal) son de tipo *Point*, es decir, puntos de luz, y están en modo *Baked.* Esto último significa que, en lugar de calcular la iluminación y sombras en tiempo real, se calcula una sola vez y se mantiene fija durante el transcurso del juego, ahorrando así una cantidad considerable de recursos computacionales.

Para disuadir al jugador de acercarse a los límites del mapa, se ha modelado el terreno de manera que las dunas más altas estén en los extremos, mientras que la zona central en la que se desarrolla la acción es bastante llana. Además, se han instalado cuatro paredes invisibles para asegurar que el jugador no caiga el vacío si intenta salirse del terreno.

En segundo lugar, se implementó un sistema de partículas dinámico que simula una tormenta de arena si el usuario se acerca demasiado al borde de la región (figura 12). De esta manera se disimula que el terreno llega a su fin y se da a entender que es mejor no acercarse a dichas zonas. Este sistema de partículas funciona según el script *Sandstorm.cs.* Cuando el jugador entra en

uno de los *triggers* cercanos a los muros invisibles del mapa, en primer lugar, se rota y desplaza el punto de origen del sistema de partículas de acuerdo con la posición actual del jugador. Se les añade cierto *offset* a las coordenadas, según de qué muro se trata, para que el jugador esté siempre en el punto más álgido de la tormenta. Por último, se comienza la emisión de partículas. Mientras el jugador se desplaza dentro del *trigger,* la tormenta le seguirá, dificultando la visión. Si el jugador sale del *trigger,* la tormenta amainará hasta desaparecer.

Figura 12. Tormenta de arena mediante un sistema de partículas.

Se han tomado además otras medidas para mejorar la sensación de inmersión si el jugador trata de acercarse a los límites del mapa. Por un lado, se editó la *skybox* mediante el software de edición de imágenes GIMP, de manera que diese la sensación de que el desierto es mucho más extenso de lo que es el terreno. Se tomó una fotografía de un desierto de licencia libre y se le aplicó un filtro impresionista para que encajase con la estética *cartoon* del juego [24]. A continuación, se editó la imagen manualmente de manera que no quedasen costuras obvias al convertir la imagen 2D en una esfera. Por último, se añadió una media luna luminiscente combinando varios efectos.

4.1.3 Cinemáticas

Aprovechando las animaciones generadas, se han realizado dos cinemáticas: una introductoria y otra conclusiva. Las cinemáticas son secuencias de eventos programados sobre las que el jugador no tiene control, como si se tratase de un vídeo. Su función es enriquecer el mundo en el que tiene lugar el juego, estableciendo la trama y la resolución de la misma. Las cinemáticas se han creado usando la pestaña de *Timeline* de Unity y un *gameObject* con el componente *Playable Director. Timeline* es una ventana de Unity que permite programar eventos que transcurrirán en determinados momentos según su posición en la línea de tiempo. Funciona con un sistema de pistas, algo similar al de un editor de audio o vídeo estándar.

Existen seis tipos de pistas o *tracks*, cada uno de los cuales permite realizar distintas las acciones que conforman una cinemática. En este caso, se han usado cuatro tipos diferentes, que se explican a continuación:

- *Animation Track*: es el tipo de pista más frecuente. Por un lado, permite controlar el componente *Animator* de un *gameObject*, de manera que se puedan reproducir las distintas animaciones presentes en su controlador de animaciones. Además, permite editar los valores del componente *Transform* del objeto, esto es, su rotación, escala y traslación. Esto último es muy útil para implementar una cámara dinámica, a lo cual se le saca partido en ambas cinemáticas. Si se quiere reproducir una animación además de editar el componente *Transform* es necesario añadir una *Override Track,* que "superpone" dos o más animaciones de un mismo objeto simultáneamente. Esto se emplea para que el camello pueda desplazarse y reproducir la animación de caminar a la vez, por ejemplo.
- *Activation Track:* permite activar y desactivar un *gameObject*, para que aparezca o desaparezca en la escena.
- *Audio Track:* permite asignar clips de audio a una determinada fuente de audio, de manera que se reproduzcan durante el tramo asignado en la línea de tiempo. Además, se pueden automatizar parámetros como el volumen o el paneo estéreo mediante curvas.
- *Signal Track:* manda una determinada señal a todos los *gameObjects* que posean un componente de receptor de señales para esa señal específica. El receptor de señales se configura con las acciones que se deben realizar si se recibe dicha señal, por ejemplo, ejecutar un método del script asociado a ese objeto. De esta manera, se pueden ejecutar partes del código en momentos determinados, sin que sea necesaria la intervención del jugador, que en este momento no tiene control sobre su personaje.

Por otra parte, el componente *Playable Director* permite controlar la reproducción de la cinemática asociada desde el código, empleando los métodos de esta clase para pausar, reproducir, consultar la duración o la posición actual del cursor de reproducción. De esta forma, se ha escrito un script llamado *SkipCutscene.cs*, que tiene dos objetivos: permitir que el usuario se salte las cinemáticas si así lo desea, y silenciar todas las fuentes de audio irrelevantes para la cinemática mientras esta se está reproduciendo.

Mientras se está reproduciendo una cinemática, si el usuario pulsa cualquier botón, aparecerá en la interfaz del juego el mensaje "Pulsa 1 para saltar la escena". Si se sigue esta instrucción, se salta al último instante de la cinemática y el jugador aparecerá en la posición inicial con pleno control sobre el personaje. Si pasan más de 5 segundos con el mensaje en pantalla, este desaparecerá y el proceso se comenzará de nuevo, siendo necesarias dos pulsaciones para saltar la escena (una para mostrar el mensaje y otra para saltarla).

La decisión de silenciar la mayoría de fuentes de audio durante las cinemáticas se tomó para priorizar la inteligibilidad y no abrumar al jugador. Si se tienen todos los sonidos de ambiente y distracción activados y a volumen alto, además de la música de las cinemáticas y la voz de los personajes, puede llegar incluso a ser molesto. Por este motivo, se optó por limitar el número de sonidos simultáneos.

A continuación, se va a comentar el contenido de cada cinemática, así como una breve descripción de lo que ocurre en cada una de las pistas de sus respectivas líneas de tiempo.

- Cinemática introductoria: esta escena tiene lugar cada vez que se inicia el juego. Al comenzar, la cámara sobrevuela el desierto de Kobarag, pasando sobre el oasis y atravesando el mercado. Finalmente llega al punto más alejado del mercado, donde se encuentra el jugador. Camilo, el camello, se sorprende al verle y se acerca para hablar con él. Le da la bienvenida al desierto y le pide ayuda. En ese momento, se gira y comienza

a caminar hacia el mercado, y la cámara, que representa la perspectiva del jugador, le sigue. Camilo comienza a explicar la premisa, presentar a los personajes y los objetivos principales del juego. Por último, desea suerte al jugador y termina la cinemática. En la figura 13 se muestra la secuenciación de estos eventos en la línea temporal de Unity. Se aprecia que se envía una señal justo al acabar la escena. Esta se encarga de reactivar las fuentes de audio silenciadas durante la cinemática

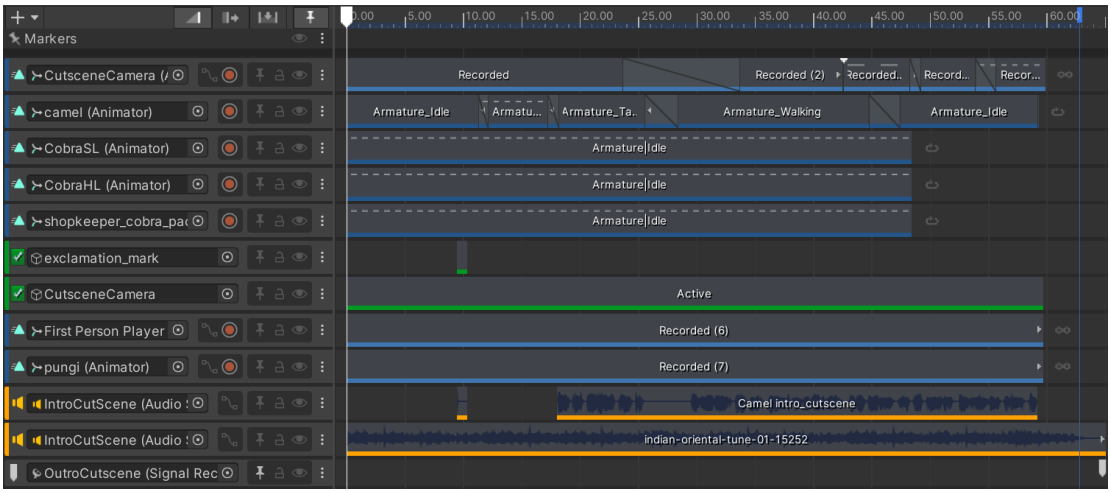

Figura 13. Ventana Timeline con la secuenciación de la cinemática introductoria.

- Cinemática conclusiva: esta cinemática ocurre tras superar todos los niveles propuestos por el terapeuta. Tras acabar el último nivel, se pregunta al usuario si quiere seguir jugando. Si acepta, se reiniciará la prueba desde el primer nivel. De lo contrario, se reproduce esta cinemática. La escena comienza con Dora y Antón, las cobras musicales, alzándose a la par que se acerca la cámara. Llegados a un punto, comienzan a bailar al son de la música. Un espectáculo de fuegos artificiales comienza en el fondo. Finalmente, las cobras descienden hasta su posición inicial. Se trata de una celebración para felicitar al jugador por completar los niveles, independientemente del resultado. Los fuegos artificiales se obtuvieron de un paquete de efectos 3D de la *Unity Asset Store*, compartido por *Creepy Cat* [25].

4.2 Estructura del juego

En este apartado se explica la secuencia de sucesos que tienen lugar en la sección principal del juego. Por "sección principal" se entiende la prueba de memoria auditiva propiamente dicha. Se comienza explicando cómo iniciar la prueba, la estructura de rondas y niveles, y las consecuencias y recompensas que se dan dependiendo de los aciertos y fallos del jugador.

Tras acabar la cinemática inicial, el jugador aparece frente a los puestos del mercado. Si se acerca a una de las cobras, esta se gira en su dirección y le pregunta si quiere comenzar a jugar (siempre y cuando el jugador esté mirando en dirección a la cobra). En este momento se deshabilitan el control de movimiento y de cámara, y aparecen dos opciones en pantalla. Si el jugador acepta comenzar una ronda, será desplazado automáticamente hacia un punto "neutral"

frente a la cobra desde el que se realiza la prueba. De lo contrario, se recupera el control del personaje.

Durante la prueba el jugador no puede desplazarse por el mapa, aunque sí mover la cámara. La cobra procede a cantar una secuencia de notas generada, y al acabar, el jugador usa su *pungi* para repetirlas. Mientras el jugador toca el instrumento, la cobra baila. Las notas que se aciertan se indican mediante círculos verdes ubicados en la esquina inferior izquierda de la interfaz de usuario, además de que aparece una exclamación sobre la cobra. En caso de fallar una nota, aparece un círculo rojo en la interfaz de usuario y una interrogación sobre la cobra.

El jugador tiene tres oportunidades para acertar el patrón de notas correcto. Si se introducen las notas acertadas, la serpiente se eleva ya que está siendo "encantada". En cambio, cada vez que se falle una nota, la cobra repite la secuencia correcta para refrescar su memoria. Si el jugador introduce una secuencia incorrecta tres veces, la cobra se desplaza hacia abajo y pasa de ronda, reproduciendo el siguiente patrón.

Se denomina "ronda" a cada una de las secuencias de notas que el jugador deberá tratar de memorizar y repetir. Un nivel es un conjunto de rondas que comparten los mismos ajustes en cuanto a la longitud de las secuencias y la velocidad de reproducción. Se pueden configurar hasta un máximo de cinco niveles desde la página web. En el modo local, mediante el menú propio del juego, se puede configurar un solo nivel.

Al acabar todas las rondas de un nivel, se pasa al siguiente (si es que lo hay). Esto se comunica mediante el rótulo de *Level Up* que aparece en pantalla. Además, la cobra tendrá una reacción de celebración o confusión dependiendo del número de secuencias correctas por parte del jugador. Si se supera el último nivel, se pregunta al usuario si quiere volver a jugar. Si acepta, se repetirá el proceso empezando desde el primer nivel. Si se rechaza la oferta, se reproduce la cinemática conclusiva, felicitando al jugador por haber completado todos los niveles. En la figura 14 se representa la estructura del juego en forma de diagrama de flujo.

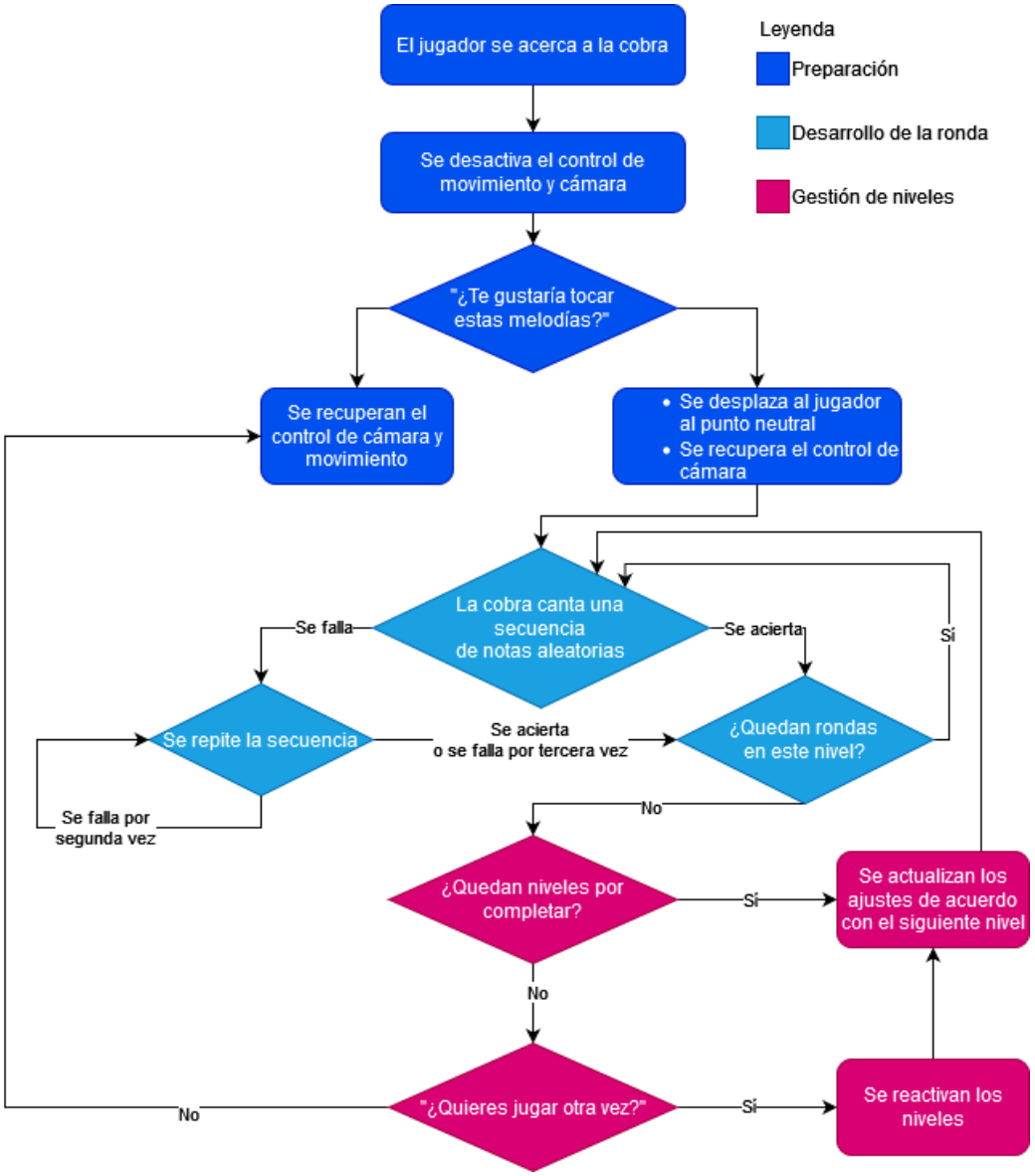

Figura 14. Diagrama de flujo de la prueba de memoria auditiva.

4.3 Lógica interna

Gran parte del tiempo dedicado a este proyecto se ha empleado en la programación del videojuego, y por tanto resulta necesario dedicar un apartado a indagar en su lógica interna. No se va a entrar en detalle en todos los *scripts* del juego, pero sí en los más importantes. Los *scripts* que manejan la sección principal del juego son *CobraRoundStart.cs*, *PatternManager.cs*, *PlayerPattern.cs* y *LevelValues.cs.* En primer lugar, conviene hacer un pequeño resumen del papel que desempeñan dentro del juego, así como de la forma en la que interactúan unos con otros.

- *CobraRoundStart.cs* permite al usuario interactuar con las cobras principales fuera de las rondas. Por tanto, su función primordial es preguntar si se quiere dar comienzo a la prueba de memoria auditiva, o si quieren repetir los niveles tras finalizarla. Si se acepta, pide a *PatternManager.cs* que cree una nueva secuencia. También comunica al resto de *scripts* con cuál de las dos cobras se está interactuando, ya que esto afecta al modo de juego.
- *PatternManager.cs* maneja todo lo referente a las rondas de memoria auditiva. Lo más relevante es creación y reproducción de patrones, así como la corrección de las notas introducidas por el jugador. Dependiendo de dicha corrección, repetirá la ronda o pasará a la siguiente.
- *PlayerPattern.cs* permite al usuario introducir notas durante las rondas o solicitar una repetición de la secuencia reproducida. Por tanto, se encarga de los controles del juego (exceptuando el movimiento y la cámara).
- *LevelValues.cs* almacena los ajustes de cada uno de los cinco niveles que se pueden rellenar desde la página web. Si se termina el nivel actual, permite pasar al siguiente, desactivando el que se acaba de superar y asignando los valores del próximo nivel a las variables oportunas.

El diagrama de flujo de la figura 15 ilustra de manera gráfica las interacciones entre las clases mencionadas. Cada uno de los *scripts* se representa con un color distinto. Los elementos del diagrama en los que existen bifurcaciones se han representado con rombos, mientras que el resto se han representado con rectángulos. En las bifurcaciones, las respuestas afirmativas son de color verde, mientras que las negativas son de color rojo.

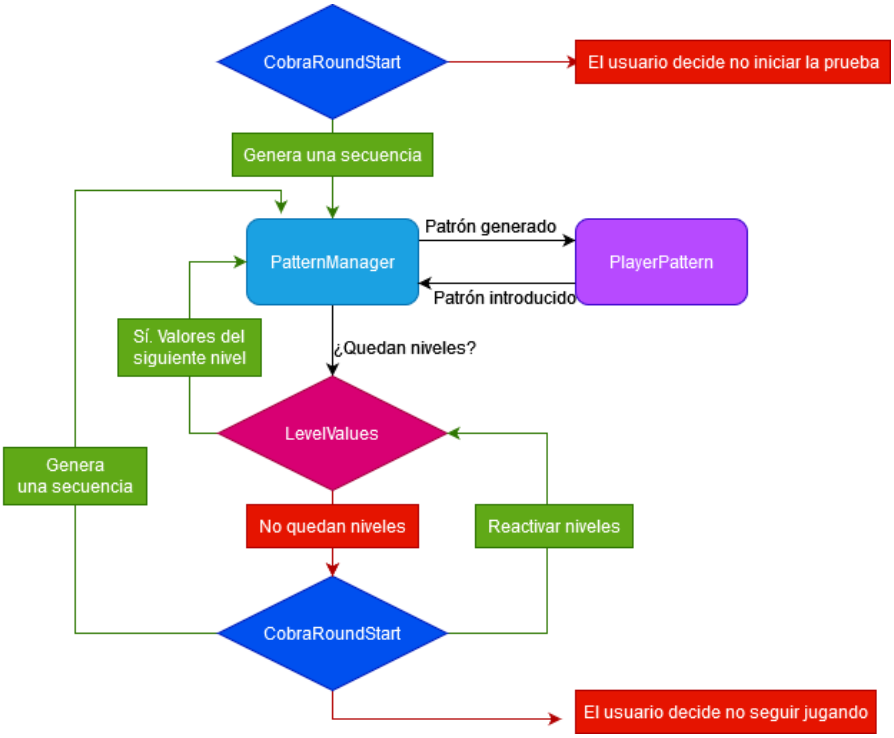

Figura 15. Diagrama de flujo de la lógica interna en la sección principal del juego.

Una vez se ha resumido y explicado la interacción entre los *scripts* que conforman la sección principal del videojuego, se va a indagar en cada uno de ellos de manera más específica, haciendo referencia a sus métodos y variables más relevantes.

4.3.1 CobraRoundStart.cs

CobraRoundStart.cs se encarga de la interacción entre el jugador y la cobra antes de empezar una ronda.

1. **Comprobaciones previas a la interacción.** Para evitar que la cobra hable al jugador de manera accidental cuando este camina próximo a ella, se ha empleado un *raycast*. Se dibuja un rayo desde la cámara del jugador que activa un booleano (*lookingAtSnake*) cuando este entra en contacto con una de las cobras, y, de esta manera, sólo se interactúa con ellas de manera intencionada. Este mismo método se usa también con el resto de personajes. Existe además otro booleano, *newRoundAvailable* que sirve, entre otras cosas, para que si el jugador rechaza iniciar una ronda no se le vuelva a preguntar inmediatamente. Por tanto, para que la cobra pregunte al jugador si quiere empezar a jugar, deben cumplirse esas dos condiciones simultáneamente.

2. **Selección del modo de juego**. Si se cumplen las condiciones mencionadas, en primer lugar, es necesario comunicar a las clases *PatternManager* y *PlayerPattern* si el jugador se encuentra frente a la cobra verde o la cobra morada. De ello depende el modo de juego que se va a iniciar inminentemente (notas largas y cortas o notas graves y agudas). Esto se realiza llamando al método *ChangeCobra* de cada una de estas clases y cambiando la variable booleana *shortLong*

de *PlayerPattern.* Esta variable es la que se consulta para saber cuál es el modo de juego actual. Si está activada, son notas cortas y largas; si está desactivada, son notas graves y agudas.

3. **Solicitud de respuesta por parte del usuario**. El siguiente paso es preguntar al jugador si desea empezar la prueba de memoria auditiva. Para ello se desactiva el movimiento del jugador y de cámara, y se entra en el método *WaitForInput.* Esta corrutina que espera indefinidamente a que el jugador dé una respuesta, y realiza las acciones oportunas una vez contesta. Si la contestación por parte del jugador es afirmativa se llama a la función *CreatePattern* de la clase *PatternManager* y se da comienzo a una ronda. Se explica su funcionamiento en detalle en el siguiente subapartado, 4.3.2 PatternManager.cs y PlayerPattern.cs.

4. **Desplazamiento del jugador**. Si el usuario acepta comenzar a jugar, se usará el método *MovePlayer* para desplazarle al punto neutral desde el que se realiza la prueba. Se trata de otra corrutina que utiliza interpolación lineal entre el punto actual del jugador y las coordenadas del punto objetivo. Este punto neutral no es más que un *gameObject* vacío situado en un punto estratégico frente a la cobra (*SLTarget* o *HLTarget* según si se trata de la cobra verde o morada). Durante el desplazamiento, la cámara del jugador deberá apuntar a la cobra en todo momento, y para ello se sigue un método similar, con *gameObjects* vacíos situados en las caras de las cobras, y calculando la rotación del jugador (y, por tanto, de la cámara) conforme se produce el desplazamiento.

4.3.2 PatternManager.cs y PlayerPattern.cs

PatternManager.cs es el *script* principal del juego, ya que gestiona todo lo relacionado con la creación y reproducción de patrones, el recuento de aciertos por parte del jugador y el sistema de rondas, así como todas las acciones que se toman en consecuencia. *PlayerPattern.cs* es el *script* que permite al usuario introducir su secuencia de notas. Como ambas clases trabajan en conjunción durante las rondas, se ha decidido que compartan el mismo subapartado en lugar de explicarlas por separado.

1. **Creación de patrones de notas**. Cuando se llama a *CreatePattern* se crea una patrón de notas de la longitud que indican los ajustes, que se almacena en la variable *pLength*. Internamente, estas secuencias son *arrays* de números enteros, donde "1" significa que la nota es corta o grave, y "2" que la nota es larga o aguda, dependiendo del valor de la variable *shortLong.* El *array* que contiene estos números enteros se llama *correctPattern*. Al final de este método, se llama a la corrutina *PlaySequence*, que es la que se encarga de reproducir el patrón generado.

2. **Reproducción del patrón generado.** En el método *PlaySequence* se recorre el *array* y se asigna el sonido correspondiente a la fuente de audio. Antes de reproducir la siguiente nota, la corrutina ha de mantenerse en espera durante un intervalo de tiempo equivalente a la duración de la nota anterior, de manera que nunca se solapen. Además, existe un parámetro llamado *pauseAfterNote*, que permite alargar el tiempo de espera entre una nota y la siguiente. Este parámetro se puede configurar desde los ajustes, ofreciendo distintas velocidades de reproducción de la secuencia de notas, de manera que se pueda ajustar a las capacidades de cada jugador.

Cada vez que se reproduce una nota se activa la animación de cantar de la cobra mediante un *trigger*. La velocidad de reproducción de la animación varía de forma proporcional a la longitud de la nota y al parámetro *pauseAfterNote*.

3. Introducción de notas por parte del jugador. Una vez se ha reproducido el patrón completo, se activa el booleano *cPatternComplete* y el *pungi* aparece frente al jugador, como indicativo de que es su turno y debe repetir la melodía. Aquí entra en juego la clase *PlayerPattern*, que se encarga de los controles del jugador referentes a tocar secuencias de notas, y almacena el patrón que introduce. Existen dos configuraciones diferentes para los controles del juego en lo que respecta a cómo el jugador introduce su secuencia de notas. Se profundiza en esto en el apartado 4.4 Controles.

Por el momento, basta con saber que la clase *PlayerPattern* permite al jugador introducir una nota o pedir una repetición de la última secuencia reproducida, siempre y cuando *cPatternComplete* esté activo. Cuando se introduce una nota, se guarda en el *array playerPattern* el número entero correspondiente (1 o 2), se llama al método *PlayNote* para que se reproduzca el sonido, se contabiliza la pulsación mediante la variable *buttonPress* y se hace una llamada al método *CompareNote* de *PatternManager.*

4. Corrección de las notas introducidas. En *CompareNote* se comprueba si cada nota que se introduce es correcta o no, y se encarga de que se muestren los *sprites* correspondientes según el caso (los puntos verdes y rojos de la interfaz de usuario, y las exclamaciones e interrogaciones que aparecen sobre las cobras). Cuando la variable *buttonPress* de *PlayerPattern* alcanza el valor de *pLength*, el jugador habrá introducido un patrón de notas de la misma longitud que el generado por el juego, y la ronda estará completa. Si se ha fallado alguna nota, se llama a *PlaySequence* para que se reproduzca de nuevo el último patrón generado, dando un máximo de tres oportunidades al jugador para acertar.

5. Desplazamiento de la cobra. Si el jugador ha acertado la secuencia, se llama al método *MoveCobra* y la cobra ascenderá una distancia determinada, denominada *cobraStep.* Cada vez que se desplaza la cobra, se guarda la distancia total respecto a la posición inicial en la variable *netDisplacement*, para que pueda volver a su posición original una vez acabado el nivel. *MoveCobra* (al igual que *MovePlayer,* de la que se habló anteriormente) es una corutina *IEnumerator* que utiliza la interpolación lineal para desplazar de manera fluida el modelo de la cobra. Si el jugador falla una secuencia de notas tres veces, la cobra en lugar de ascender bajará. Sin embargo, para evitar que la cobra se entierre, esto solo será posible si *netDisplacement* es, al menos, mayor que *cobraStep*. Al completar un nivel, se inicia la corrutina *MoveCobra* pero esta vez con la distancia *netDisplacement* cambiada de signo, de manera que la cobra retorna a su altura neutral.

6. Suma de gemas. Al acabar una ronda se hace una llamada a *AddGems*, que se encarga de sumar la cantidad correspondiente de gemas según las respuestas dadas por el jugador. Si además de terminarse la ronda se ha completado un nivel, se añadirán las gemas de bonificación apropiadas. En el apartado 4.5 Motivación del jugador se indaga en el sistema de gemas y recompensas.

7. Repetición de niveles. Si se han completado todos los niveles, se llama al método *RepeatLevels* de *CobraRoundStart*, que pregunta al jugador si quiere volver a jugar, es decir, empezar de nuevo desde el primer nivel. Este proceso hace uso de la corrutina *WaitForInput* mencionada anteriormente. La gestión de niveles se realiza mediante un script llamado *LevelValues*.

4.3.3 LevelValues.cs

LevelValues.cs es el script que se encarga de almacenar los ajustes de cada uno de los cinco niveles configurables desde la web. Por tanto, por cada parámetro configurable en un nivel (como podría ser la longitud del patrón, por ejemplo) esta clase tiene un array de cinco elementos en el que la posición 0 serían los ajustes para el nivel 1 y la posición 4 los ajustes para el nivel 5.

1. Sistema de niveles activos. Para que los ajustes de un determinado nivel se consideren válidos, el nivel debe estar "activado". Un nivel está activado cuando el booleano del *array levelActive* de la posición correspondiente a ese nivel es verdadero. Cuando un nivel activo es completado por el jugador, este se desactiva. No obstante, al iniciar el juego los valores originales de *levelActive* se guardan en otro *array* equivalente, *originalLevelActive.* De esta manera, si el jugador decide volver a jugar tras completar los niveles que estaban activos, el método *ResetActiveLevels* puede reactivarlos consultando los valores de *originalLevelActive.*

2. Pasar al siguiente nivel. Otros métodos útiles de esta clase son: *LevelsLeft,* que devuelve un booleano según si quedan niveles activos por completar; *EndCurrentLevel,* que desactiva el nivel actual tras completarlo y, por último, *ObtainValues*, que asigna los valores del siguiente nivel activo a las variables correspondientes para que entren en vigor esos ajustes. Por tanto, estos tres métodos trabajan en conjunción para pasar al siguiente nivel: se realiza la comprobación de *LevelsLeft* y, de ser positiva, se desactivaría el nivel actual (*EndCurrentLevel*) y se obtendrían los valores del siguiente (*ObtainValues*).

4.4 Controles

Un elemento fundamental de cualquier videojuego es que los controles sean intuitivos. El jugador debería poder asimilar la función de cada botón tras un periodo de adaptación relativamente corto. Lo ideal es que el mando o teclado sea un intermediario "invisible", que no añada dificultad a la hora de que el jugador ejecute en el juego las acciones que tiene en mente.

Tal y como se ha establecido, el objetivo principal de este juego es reproducir las secuencias de notas solicitadas. A la hora de plantear cómo podría el jugador introducir notas de distinta duración de manera intuitiva se idearon dos soluciones. Se va a comenzar explicando la más sencilla a nivel lógico, y posteriormente se hablará de la más compleja.

4.4.1 Modo "Dos botones"

La primera opción es la más básica: existen dos tipos de notas (largas y cortas), por lo que se usan dos botones distintos, cada uno asignado a un tipo de nota. De esta forma, si se pulsa el primer botón, se registra una nota corta y se reproduce un archivo de audio de corta duración. Si se pulsa el segundo botón, se registra una nota larga y se reproduce un archivo de audio de larga duración.

Esta configuración de control resulta muy sencilla de implementar, pero no es especialmente intuitiva. Es posible que el jugador tarde un tiempo en asociar qué botón se corresponde con cada tipo de nota. Como ayuda, debe aparecer una leyenda en la interfaz del juego en todo momento.

4.4.2 Modo “Un botón”

La segunda solución consiste en un solo botón, que, dependiendo de la cantidad de tiempo que se pulse, registra una nota corta o larga respectivamente. Mientras el botón esté siendo pulsado, se cuantifica la duración de la pulsación. Si supera un determinado límite, se considera que la nota es larga. De lo contrario, se trata de una nota corta. Este límite, tras realizar varias pruebas con distintas personas, se estableció en 300 milisegundos. Mientras el botón esté siendo pulsado, el instrumento deberá seguir sonando, ya que pulsar el botón en este caso es análogo a soplar un instrumento de viento. Los archivos de audio que se reproducen durante la pulsación se nombraron con el prefijo *single_note*, es decir, “nota única”. Esto se debe a que en este modo de juego se reproduce siempre el mismo archivo, independientemente de si se introduce una nota corta o larga.

La dificultad de la implementación de este modo surge de que se tienen que ejecutar distintas instrucciones en el instante que se pulsa el botón (en Unity esto se conoce como *KeyDown*), mientras se mantiene pulsado (para ello se creó el estado *isBeingPressed*) y al dejar de pulsar el botón (*KeyUp*). El algoritmo elaborado finalmente se muestra en la figura 16.

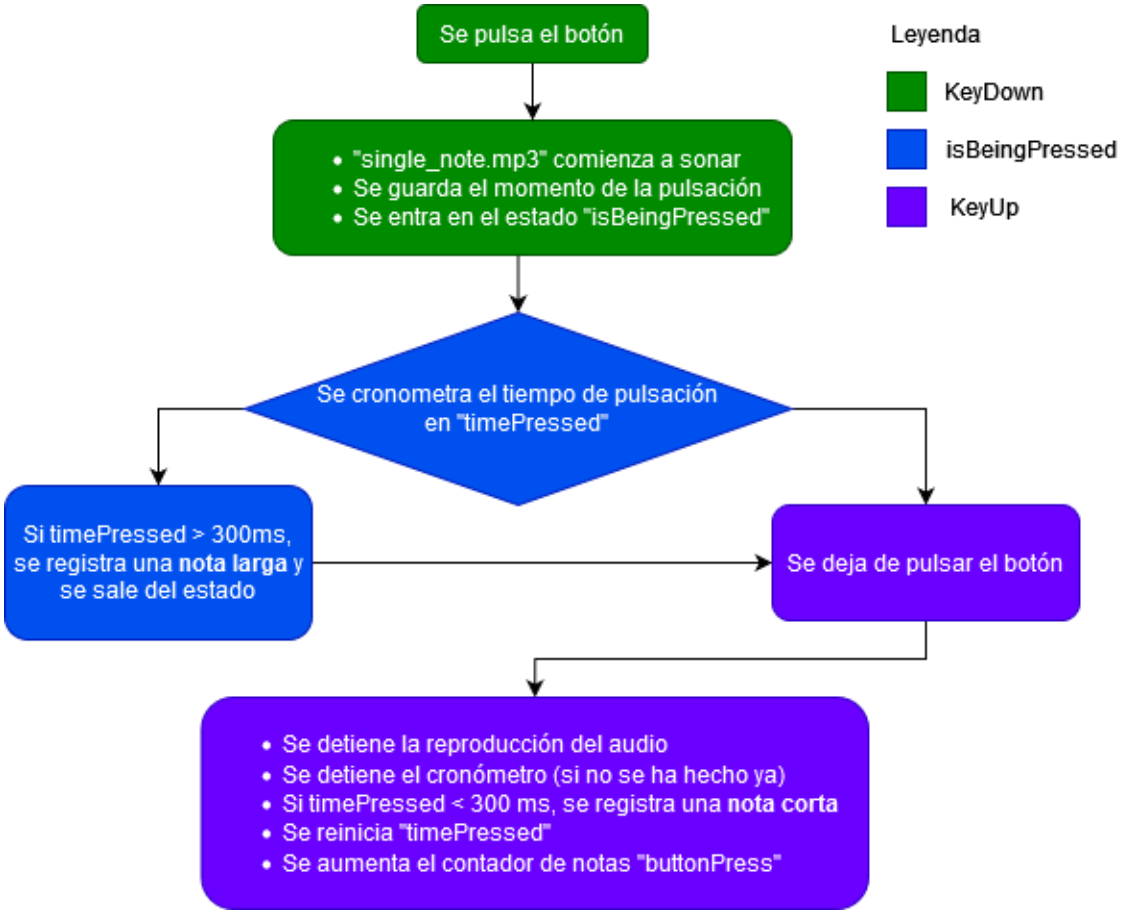

Figura 16. Diagrama de flujo del modo de control "Un botón".

Desde el momento de la primera conceptualización del juego, estos dos esquemas de control se han valorado como opciones viables, y finalmente se decidió mantener ambos a petición de una de las terapeutas de Aura. El modo "un botón" es más intuitivo, ya que el usuario puede traducir de manera directa la secuencia de notas que ha escuchado en el movimiento de pulsación, sin tener que identificar cada una de las notas como "corta" o "larga" de manera consciente. El modo "dos botones" obliga usuario a memorizar la secuencia, examinar la duración de cada una de las notas individualmente, categorizándolas como "corta" o "larga" y por último asociar esta categoría con el botón correspondiente. Este último modo, a pesar de ser más difícil para la mayoría de jugadores, se asemeja más a los exámenes que realizan las audiólogas. Por tanto, la inclusión de ambos modos de control está justificada.

Tras haber implementado la prueba de memoria auditiva con notas de distinta duración, se añadió un segundo modo de juego, con notas graves y agudas. En este caso, la configuración de control de un solo botón no tiene sentido (la duración de la pulsación difícilmente puede asociarse al tono de la nota), por lo que se usa siempre el modo "dos botones". Al interactuar con la cobra morada, se guarda la configuración de control actual y se cambia automáticamente al modo "dos botones" Una vez se han completado todos los niveles y se recupera el control del personaje, se devuelve la configuración de control elegida en los ajustes originalmente.

4.4.3 Implementación de control por mando

Uno de los objetivos que se estableció para este proyecto es que el juego pudiera controlarse tanto con teclado y ratón como con mando. Afortunadamente, Unity ofrece al desarrollador herramientas que facilitan esta implementación.

Por ejemplo, para el movimiento del personaje, si se usa el método *GetAxis* de la clase *Input* se pueden asignar de manera automática los ejes vertical y horizontal a las teclas WASD del teclado y al joystick izquierdo del mando. Esto se debe a que los ejes "Horizontal" y "Vertical" vienen predefinidos en el editor y asignados a estas entradas por defecto. El movimiento del personaje en el juego funciona mediante el *script PlayerMovement.cs,* inspirado en gran medida por el tutorial de Brackeys [26], y aprovecha estos métodos y ejes predefinidos por Unity.

En cuanto a la cámara, está implementada mediante el *script FirstPersonCamera.cs*, basada en el tutorial de Unity Ace [27]. En este caso se vuelve a usar *GetAxis* con los ejes "Mouse X" y "Mouse Y". Estos ejes también vienen predefinidos y asociados al movimiento del ratón, como su nombre indican. Sin embargo, para que la cámara funcione también mediante el mando será necesario redefinirlos en la ventana *Input Manager* del editor. Los ejes "Mouse X" y "Mouse Y" se duplican y se asocian al "joystick axis 4" y "joystick axis 5" respectivamente. Por último, se configuran la sensibilidad, gravedad y la zona muerta del joystick.

Asignar funciones a los botones A, B, X e Y del mando de Xbox es bastante directo. Estos botones a nivel interno se reconocen como "joystick button" seguido de 0, 1, 2 o 3 respectivamente. Basta con añadir la condición de que se pulse el botón deseado del mando a los métodos que buscan *input* del teclado. Por ejemplo, si se quiere que la tecla 1 del teclado y el botón A del mando cumplan la misma función, la condición "`if((Input.GetKeyDown(KeyCode.Alpha1))`" pasaría a ser "`if((Input.GetKeyDown(KeyCode.Alpha1) || Input.GetKeyDown("joystick button 0"))`". La función específica de cada botón del mando en el juego se explica en detalle en el Anexo II: Manual.

El último obstáculo por resolver era que los botones que parecen en la interfaz del juego se actualizasen automáticamente si se detecta que hay un mando conectado. Para ello se creó el script *ControllerSupport.cs*. En primer lugar, se implementó el método *ControllerConnected*, que devuelve un booleano verdadero o falso dependiendo de si hay un mando conectado o no. Para ello, se consulta la lista de nombres de mandos detectados por Unity. Si la lista tiene longitud nula, o sus elementos están vacíos, no hay ningún mando conectado. De lo contrario, sí lo habrá y en consecuencia será necesario cambiar los elementos de la interfaz que hacen referencia al teclado.

Para ello, en primer lugar, se programó el método *AdjustCanvasToController.* Este método sustituye los iconos de teclas que aparecen en el *Canvas* (es decir, las indicaciones de controles durante las rondas o las preguntas de las cobras) por iconos de los botones del mando. Por otro lado, se escribió un método *GetPrompt* que devuelve el icono indicado (teclado o mando) para cada objeto o personaje con el que se interactúa (el pozo, el cofre, el camello y la cobra tendera). La comparativa entre las interfaces con teclado o mando se muestra en la figura 17.

Figura 17. Comparativa de la interfaz con teclado (izquierda) y mando (derecha).

4.5 Motivación del jugador

Un elemento indispensable en cualquier videojuego es establecer una serie de objetivos y recompensas que motiven al usuario a seguir jugando y mejorar sus habilidades. Para conseguir esta gratificante sensación de progreso se ha implementado un sistema de gemas, que son una divisa ficticia dentro del juego.

Por cada nota acertada, el jugador gana una gema. De esta manera, se valora más una respuesta errónea pero cercana a la correcta que una respuesta completamente dispar. Además, si se completa un nivel sin errores, se añade a lo anterior una bonificación de 6 gemas extra. Si el nivel se supera habiendo fallado una sola vez, la bonificación será de 3 gemas. Para niveles en los que el jugador ha introducido más de dos patrones erróneos no habrá bonificación.

Esta moneda se puede emplear en el juego para desbloquear nuevos *pungis*, alterando el aspecto y sonido del instrumento estándar. A continuación, se listan todos los instrumentos desbloqueables en el juego:

- *Pungi* de madera: es el instrumento estándar del juego, con el que se empieza al inicio de cada partida. Su aspecto es similar al de un *pungi* tradicional y suena como una flauta de madera.

- *Pungi* cactus: para desbloquearlo, es necesario lanzar una gema al pozo del mercado y pedir un deseo. Esto hace que aparezca un cofre al otro lado del oasis, en el cual se puede encontrar este instrumento. El proceso de obtención puede parecer algo complejo, pero es una especie de *easter* egg para recompensar la exploración del mapa y la interacción con todas las mecánicas del juego. Además, su bajo precio, de una sola gema, lo compensa. Tiene aspecto de cactus y suena como una guitarra eléctrica distorsionada.
- *Pungi* dorado: se obtiene en el mercado, comprándolo a la cobra tendera a cambio de 60 gemas. Su aspecto es metálico y dorado, y suena como varios instrumentos de viento metal al unísono.
- *Pungi* diamante: este instrumento es la recompensa final para los jugadores más dedicados. Para obtenerlo es necesario haber obtenido los todos los instrumentos anteriores. Solo entonces el camello lo pondrá en venta a cambio de 100 gemas. Su material es un cristal brillante y azulado y suena como un sintetizador.

Todo lo relativo al desbloqueo y cambio de *pungi* se controla mediante el script *PungiManager.* Cuando se desbloquea un nuevo instrumento, se le asigna el siguiente número disponible en la lista (entre el 2 y el 4), se aumenta el contador de *pungis* desbloqueados y se activa un booleano que indica que ese instrumento está disponible para su uso. Por último, se utiliza el método *ChangePungi* de esta misma clase para que el nuevo instrumento aparezca el primero cuando se inicie una nueva ronda y que así el jugador pueda probarlo. *ChangePungi* asigna las cinco pistas de audio propias de cada uno de los instrumentos a sus variables correspondientes y cambia el "material" (*MeshRenderer*) del *Pungi.* Por tanto, el modelo 3D es siempre el mismo, pero cambia su textura y los sonidos que reproduce.

4.6 Diseño de sonido

Al ser este un juego de entrenamiento auditivo ha sido necesario prestar especial atención al apartado de diseño de sonido durante su desarrollo. Las fuentes de audio más importantes del juego son los *pungis*, las cobras principales, los ruidos de fondo y los ruidos de distracción.

3.6.1 Audio de los *pungis*

Como se ha mencionado en el apartado anterior, existen cinco archivos de audio relativos a cada uno de los instrumentos que pueden ser usados por el jugador. Cada opción del modo "dos botones", para cada uno de los dos modos de juego existentes, tiene un archivo de audio asociado: *short_note*, *long_note*, *low_note* y *high_note.* Los archivos *short_note* y *long_note* son versiones de distinta duración del mismo archivo original. La razón por la que se han editado dos archivos diferentes en lugar de simplemente variar la duración de reproducción de un mismo archivo es para poder introducir un efecto de desvanecimiento progresivo (*fade out)* y que el audio no se corte de manera abrupta.

Por su parte, *low_note* y *high_note* tienen una duración intermedia entre las notas cortas y las notas largas. Además, se ha transpuesto su tono para que estén, respectivamente, una octava por debajo y una octava por encima del archivo original que se usa en el modo de juego de notas de distinta duración. De esta manera, con dos octavas de diferencia entre el tono de un archivo

y otro, la diferencia queda suficientemente clara independientemente del conocimiento musical del usuario.

El quinto archivo se denomina *single_note.* Se trata de una versión del archivo original aún más larga que *long_note*, ya que su propósito es que suene de manera "indefinida" mientras el jugador mantenga pulsado el botón en el modo "un botón".

En cuanto a los sonidos seleccionados, todos ellos son muestras de audio obtenidas en *freesound.org* [28] [29] [30], exceptuando el sintetizador del *pungi* diamante, que se corresponde al *preset Fake Sync Lead* de Kontakt [31].

4.6.2 Audio de las *cobras*

En la fase de conceptualización del juego, tras valorar varias opciones sobre cómo se le podría indicar al jugador qué melodía debe repetir, se decidió que sería la propia cobra la que la cantase. Esto suponía, o bien encontrar en internet grabaciones de voz con licencia gratuita que encajasen con el concepto del juego, o encontrar un cantante dispuesto a grabarlas.

Finalmente, se encontró una tercera opción. AlterEgo es un software de síntesis de voz gratuito con bancos de voz de cantantes virtuales [32]. Este plugin permite escribir una oración y asociarle una melodía escrita en *Musical Instrument Digital Interface* (MIDI) desde una *Digital Audio Workstation* (DAW), en este caso, Reaper [33]. El banco de voz seleccionado fue *Bones*, por ser el más versátil y ser el único con opción de pronunciar en inglés (los demás se limitaban a la pronunciación propia del japonés). Como texto para la síntesis de voz se recurrió a algo neutral, simplemente una entonación de la vocal "a". El último paso fue escribir mediante MIDI notas de distinta duración y tono para que fueran "cantadas", y exportar dichos archivos de audio.

Antes de añadir doblaje de voz al juego, se consideró la opción de que las cobras cantasen palabras de ánimo al jugador mediante este software. Sin embargo, el resultado dejaba mucho que desear al no estar diseñado para textos en castellano. La figura 18 muestra la interfaz del plugin, que permite introducir los textos para que sean "cantados".

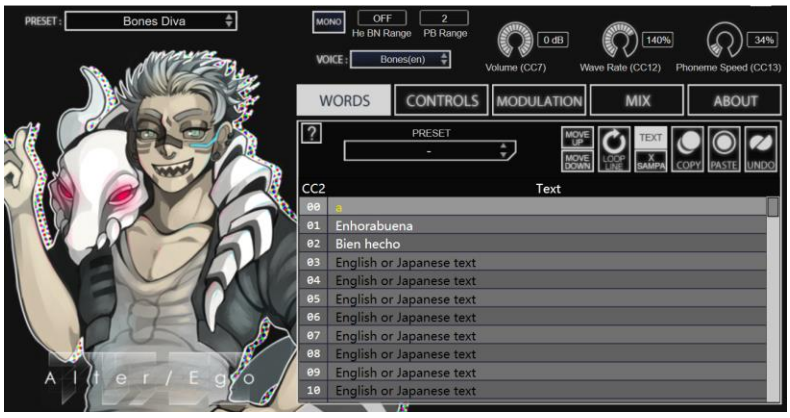

Figura 18. Interfaz del plugin AlterEgo.

4.6.3 Ruido de fondo

Uno de los mayores desafíos a los que se enfrentan los niños diagnosticados con DPAC es el procesamiento auditivo en ambientes ruidosos. Por tanto, dado que este juego pretende servir como entrenamiento auditivo, resulta indispensable que se incluya algún tipo de ruido de fondo que les ayude a practicar este aspecto de su vida diaria. El juego consta de dos tipos de ruido de fondo: el murmullo de un lugar abarrotado y el sonido del agua en movimiento. Estos ruidos además resultan inmersivos ya que el juego transcurre en un mercado junto a un oasis.

Estos ruidos de fondo debían reproducirse en bucle, sin cortes notorios que rompieran la inmersión. Para ello se editaron los archivos de audio en Audacity. En primer lugar, se cortó un extracto del inicio del audio. Este fragmento se pegó en otra pista del editor, de manera que el final del extracto y del audio completo acabasen en el mismo instante. Por último, se aplicó un efecto de desvanecimiento progresivo (*fade out)* a la primera pista, y un efecto de aparición progresiva (*fade in)* al extracto que se encuentra en la segunda pista. Siguiendo estos pasos, el nivel del audio se mantiene constante, y aparentemente comienza y termina en el mismo punto, por lo que se puede reproducir en bucle fácilmente sin que se noten cortes. Se representa gráficamente este proceso en la figura 19.

Figura 19. Procedimiento de edición de audio para una reproducción en bucle sin cortes.

4.6.4 Ruidos de distracción

Los ruidos de distracción tienen una función similar a la del ruido de fondo: dificultar la audición y procesamiento del patrón de notas cantado por la cobra. La principal diferencia es que mientras que los ruidos de fondo se reproducen constantemente y en bucle, los ruidos de distracción son sonidos breves que se reproducen de manera aleatoria pero regular, dentro de un rango de tiempo establecido.

El script que se encarga de ello es *DistractionNoise.cs.* Este script es esencialmente un temporizador que, cada vez que reproduce un ruido de distracción, escoge un tiempo de espera entre 15 y 30 segundos hasta la próxima reproducción. Existen dos tipos de ruidos de distracción implementados en el juego: el graznido de un buitre y el sonido de grillos. Estos dos sonidos pueden activarse o desactivarse en el menú de manera independiente.

4.7 Menú de ajustes y página web

Como se ha comentado anteriormente, existen dos formas de configurar los ajustes de este videojuego. La manera principal de uso va a consistir en que el terapeuta realice los ajustes desde la página web del juego. El paciente debe introducir su nombre de usuario y contraseña en el inicio del juego global. De esta forma, se descargarán de la base de datos los ajustes específicos que su terapeuta haya considerado oportunos. También se da la opción de iniciar el juego sin nombre de usuario para fines de demostración o casos excepcionales, como usuarios que carecen de conexión a internet. En este caso, el ejecutable funciona en modo local y los ajustes se realizan mediante el menú principal implementado en el propio juego.

4.7.1 Menú de configuración

Los parámetros configurables en el juego son los mismos independientemente de si los configura el terapeuta desde la página web o el usuario desde el menú interno del juego. Por tanto, conviene, en primer lugar, explicar cada uno de los ajustes que se pueden efectuar desde ambos menús:

- Número de notas: longitud de las secuencias de notas del nivel. Puede ser de 2 a 9 notas.
- *Random*: si esta *checkbox* se activa, se aleatoriza la longitud de cada secuencia de notas dentro de un rango de ±1 respecto al número de notas seleccionado. Por ejemplo, si se selecciona 4 como número de notas y se activa la opción de aleatorizar, las secuencias generadas por el juego pueden ser de 3, 4 o 5 notas.
- Velocidad: este parámetro en realidad controla la longitud de las pausas entre la reproducción de cada nota, de manera inversamente proporcional: a mayor velocidad, menor pausa. La pausa mínima es de 0.1 segundos, y la pausa máxima de 1.5 segundos.
- Rondas: el número de patrones de notas que deberán repetirse en un nivel. Pueden seleccionarse entre 1 y 5 rondas por nivel.
- Controles: se puede seleccionar el modo “un botón” o el modo “dos botones”, explicados en el apartado de 4.4 Controles.
- Sonidos: permite elegir entre “Natural”, en el que cada instrumento tiene un sonido propio y la cobra canta las notas con una voz sintética, o “Tonos Puros”, en la que los sonidos de la cobra y todos los instrumentos son sustituidos por tonos puros.
- Volumen: se pueden configurar los niveles del instrumento, cobra, sonidos de ambiente y ruidos de distracción en el rango de -80 dB a 0 dB.
- Paneo: los sonidos de las cobras, ambiente y distracciones se pueden panear de manera que se reproduzcan únicamente por el auricular derecho o izquierdo si así se desea. Todos los canales de audio tienen opción de pre-escucha, para probar los ajustes de nivel y paneo antes de empezar el juego.
- Ruidos de distracción: dos *checkbox* para activar o desactivar de manera individual los dos ruidos de distracción presentes en el juego: el buitre y los grillos.

4.7.2 Modo local

El modo local se refiere a los casos en los que el jugador no inicia sesión o carece de conexión a internet y por tanto no se pueden descargar los ajustes de la web. Si se inicia el juego en modo *offline*, se cargará la escena *KOBA-Menu.* La UI (*User Interface*) está compuesta por distintas barras deslizantes o *sliders,* menús desplegables o *drop-down* y casillas o *checkboxes.* El script que contiene los métodos necesarios para interactuar con todos estos elementos del menú se llama *MenuController.cs.* Sin embargo, los valores de cada uno de los parámetros se guardan en variables de otro script, *TitleScreenMenu.cs.* La decisión de separar en dos scripts diferentes la interacción con la interfaz y los ajustes establecidos se debe a que el objeto con contiene los valores (es decir, *TitleScreenMenu*) se guardará mediante el método *DontDestroyOnLoad.*

Al cargar una nueva escena, se destruyen todos los objetos presentes en la escena actual. Mediante la instrucción *DontDestroyOnLoad*, se pueden conservar los objetos necesarios tras la transición. De esta forma, los ajustes del menú de inicio contenidos en las variables de *TitleScreenMenu* serán accesibles desde la escena principal del juego. Este objeto es identificable mediante la etiqueta "Settings".

Para cargar la escena principal del juego una vez hechos los ajustes es necesario pulsar el botón "JUGAR". Al pulsar este botón, además de cargar la escena *KOBA-Desert,* se activa un booleano llamado *playButtonClicked*. Si al iniciar la escena principal este booleano está activo, significa que se ha accedido desde el menú local, y, por tanto, se buscará el objeto con la etiqueta "Settings" que contiene los ajustes establecidos por el usuario. En la figura 20 se muestra el menú de ajustes del juego, esto es, la escena *KOBA-Menu*.

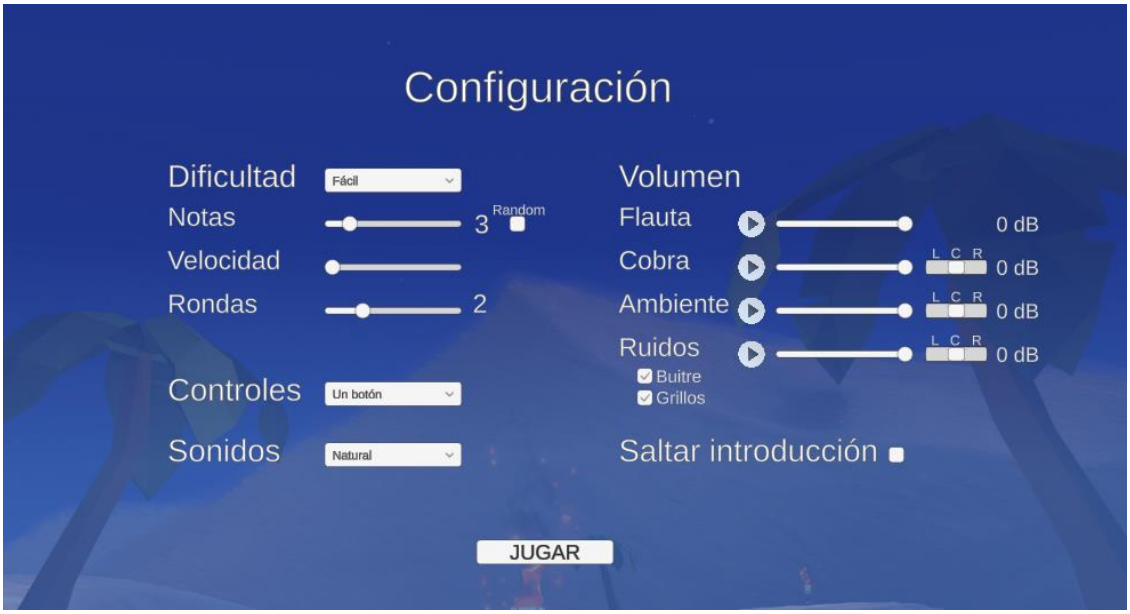

Figura 20. Menú principal del juego en modo local.

Existen algunas ligeras discrepancias entre el menú local y el menú de la web. En primer lugar, desde el menú local se puede configurar un solo nivel, frente a los cinco de la web. También se retiró de la página web la opción de omitir la introducción desde el menú, ya que puede hacerse pulsando un solo botón al comenzar la cinemática.

Por último, sólo en el modo local existe un menú desplegable que permite seleccionar dificultad "Fácil", "Medio" o "Difícil". Cambiar la opción del menú resulta en distintos ajustes predeterminados en los parámetros de Notas, Velocidad y Rondas. Finalmente se descartó esta idea para el menú de la web, ya que debe ser el terapeuta el que ajuste la dificultad, y su criterio puede diferir de los ajustes predeterminados. Además, cada paciente tiene unas necesidades diferentes. En el modo local finalmente se mantuvo, ya que para sesiones de juego rápidas facilita la configuración de varios ajustes con un solo clic.

4.7.3 Modo online

Como se ha comentado al inicio del apartado, el juego ha sido diseñado para que se use principalmente configurando los ajustes a través de la interfaz web. Cuando un jugador inicie el juego, deberá introducir su nombre de usuario y contraseña. Al iniciar sesión, se envía una petición al servidor web de "El Planeta de Sonoa", que contiene una base de datos en la que se guardan los ajustes de cada juego para cada usuario. El servidor devuelve en formato JSON las configuraciones que existen para el usuario que acaba de iniciar sesión. En el caso de Kobarag, estos ajustes se guardan en el script *LevelValues.cs* (componente de un *gameObject* llamado *WebSettings*). La escena *KOBA-Menu*, en la que se encuentra el menú local, se omite, y se carga directamente *KOBA-Desert,* la escena en la que transcurre el juego. La interfaz web, que permite hacer cambios en la base de datos, se muestra en la figura 21. Se puede apreciar que consta de cinco niveles, frente al nivel único del modo local.

Figura 21. Interfaz de usuario del menú de configuración de la web.

4.7.4 Resultados guardados

La página web, además de permitir configurar los ajustes de juego de cada paciente, también permite ver los resultados de las sesiones de juego anteriores. Cada vez que se finaliza una ronda, se añaden una serie de parámetros relevantes a sus respectivos *arrays* dinámicos. Cuando el usuario sale del juego, los datos recopilados se guardan en un fichero de formato JSON que se envía a la base de datos. La web accede a la información de la base de datos y presenta los resultados en una interfaz ordenada y limpia, que facilita su análisis. Los parámetros que se guardan para cada una de las rondas jugadas son los siguientes:

- La hora a la que se empezó a reproducir la secuencia de notas de la cobra.
- La hora a la que se terminó de rellenar el patrón de notas del jugador.
- El patrón de notas producido por la cobra.
- El patrón de notas introducido por el jugador.
- La velocidad de reproducción de la secuencia de la cobra.
- El número de intentos fallidos previos para la secuencia de notas actual.
- El número de veces que el jugador ha solicitado que se vuelva a reproducir el patrón de notas.
- El modo de juego (notas cortas y largas o notas graves y agudas).
- El modo de control del juego (un botón o dos botones).

Guardando estos parámetros se podrían analizar distintos aspectos y estadísticas acerca de cada usuario. Por ejemplo, si le cuesta más la discriminación basada en la duración o el tono, o el modo de un botón o dos botones. También podría evaluarse su progreso a lo largo del tiempo, por ejemplo, analizando si cada vez tarda menos en introducir las notas o necesita menos intentos para acertar.

5. Impacto ambiental y social

En todo proyecto de ingeniería resulta esencial tener en cuenta el efecto que este puede tener en la sociedad y el medioambiente. Es necesario hacer un balance de los aspectos positivos y negativos de la innovación de manera objetiva. En especial, porque puede darse el caso de que, en el intento de suplir una carencia, se genere un nuevo problema que no se había considerado. En los próximos subapartados se explora el impacto ambiental y social de este trabajo.

5.1 Impacto ambiental

El impacto medioambiental de los videojuegos no es necesariamente trivial. Por ejemplo, el uso regular de un PC *gaming* de alta categoría puede llegar a consumir 1100 kWh al año, más del doble que un electrodoméstico en ese mismo periodo. Por otro lado, existe un aumento en la demanda de videojuegos retransmitidos en tiempo real desde la nube, o experiencias con multijugador *online* que atraen a millones de usuarios simultáneamente. Estos tipos de videojuegos requieren una enorme infraestructura formada por numerosos servidores que consumen mucha más electricidad que los juegos en los que el procesamiento se lleva a cabo localmente [34].

Afortunadamente, este videojuego no requiere de un ordenador con altas especificaciones técnicas. Los modelos *3D* empleados son de pocos polígonos y nunca se ejecutan demasiadas instrucciones simultáneamente, por lo que es sencillo a nivel computacional. Por otro lado, aunque sí que se obtienen los ajustes de una base de datos online, se trata de una única petición, por lo que el uso de servidores es mínimo, y más aún estando este videojuego dirigido a un público de nicho. Este *software* se puede distribuir digitalmente, por lo que el empaquetado no sería necesario. Por tanto, no se genera ningún tipo de residuo y la vida del producto podría ser potencialmente indefinida.

5.2 Impacto social

“El Planeta Sonoa” es un proyecto con un carácter social intrínseco ya que su objetivo principal es ayudar a niños que diagnosticados con DPAC. Para el público objetivo del juego, menores de 5 a 14 años, tener que asistir regularmente a sesiones de terapia basadas en repetir determinados ejercicios podría llegar a ser tedioso. Con este videojuego se pretende amenizar este entrenamiento auditivo, de manera que sea más divertido y motivador.

Este proyecto nace de la ausencia de recursos en castellano especializados en este trastorno, los cuales eran muy demandados por los terapeutas. Tal y como se mencionó en el apartado 2. Antecedentes y marco tecnológico, durante el desarrollo de este juego se realizaron varias reuniones con audiólogas del Centro de Audiología y Logopedia Áurea TAV Madrid. En estas conversaciones se comentó cómo las pruebas de discriminación auditiva de notas largas y cortas generalmente se realizaban en el centro mediante un audiómetro que emite tonos puros, o incluso tocando una flauta dulce.

Este juego consta de una gran cantidad de opciones de personalización de las que carecen las herramientas usadas actualmente. Por ejemplo, consta de un mezclador de audio totalmente configurable, y los ruidos de fondo y de distracción están incluidos en el juego, por lo que no tendrían que reproducirse a través de un medio externo. Además, mediante el sistema de niveles se puede configurar toda una sesión de terapia de manera que la dificultad aumente

progresivamente de manera automática. Otro elemento diferenciador de gran importancia es que mediante la página web se pueden configurar sesiones a distancia, de manera que el paciente juegue en su casa con los ajustes establecidos por el audiólogo.

En conclusión, “El Planeta Sonoa” podría tener un efecto positivo tanto en la vida de los pacientes como en la de los terapeutas. Además, su impacto no es meramente teórico, ya que se podría empezar a usar en el Centro Áurea TAV de manera prácticamente inmediata. El balance general del impacto social y ambiental del proyecto es claramente favorable, ya que se puede considerar que no tiene ningún efecto negativo, pero sí un impacto positivo notable.

6. Conclusiones

En primer lugar, se van a revisar los puntos listados en el apartado 3. Objetivos del proyecto. El módulo consta de dos modos de juego, es decir, dos pruebas de entrenamiento auditivo. Una de ellas se centra en la diferenciación de notas de larga y corta duración, mientras que la segunda se centra en la discriminación auditiva de notas de distinta frecuencia. Estas secuencias de notas se generan aleatoriamente y el jugador tendrá hasta tres intentos para repetirlas correctamente. Este es el objetivo central del proyecto, y, en cierto modo, el más sencillo. El verdadero reto fue convertir este concepto simple en una experiencia llamativa para los usuarios y de gran utilidad para los terapeutas.

Se ha tenido en cuenta en todo momento que el juego sea divertido y atractivo para el público objetivo. Para que el juego resulte vistoso se han diseñado personajes entretenidos con numerosas animaciones y doblaje de voz, se ha escrito un guion que aporta personalidad a cada uno de ellos y se han creado cinemáticas. Para motivar al jugador se ha implementado el sistema de gemas y obtención de instrumentos. Estas recompensas marcan metas que impulsan al jugador a seguir intentándolo y repetir los ejercicios tantas veces como sea necesario.

Existen una gran variedad de parámetros configurables en el juego, permitiendo adaptar la experiencia a las necesidades de cada usuario. Simplemente variando el número de notas de la secuencia, la velocidad de reproducción o el número de rondas se puede dar lugar a experiencias de juego completamente distintas. A esto se suma las dos modalidades de control, la capacidad de sustituir los sonidos por tonos puros y un mezclador de audio muy completo y personalizable.

Se han diseñado distintas modalidades de control con la intención de hacer el juego accesible a todo tipo de personas independientemente de su experiencia previa. Por un lado, se puede elegir en los menús la manera de introducir notas de distinta duración, mediante el modo “un botón” o “dos botones”. Además, se ha prestado especial atención a la compatibilidad con mando de Xbox, ajustando todos los iconos de la interfaz si se detecta que hay uno conectado.

Por último, se ha conseguido diseñar un menú dentro del propio juego que permita configurar los parámetros en modo local. Al mismo tiempo, se han realizado los cambios oportunos para facilitar la conectividad con la página web y la integración del módulo en el proyecto principal de “El Planeta Sonoa”.

Finalmente, se van a compartir algunos de los contratiempos y reflexiones que han surgido durante el desarrollo de este videojuego. El mayor obstáculo durante la elaboración del proyecto ha sido la organización temporal. El proceso de desarrollo de un videojuego es altamente imprevisible y, por tanto, difícil de planear. En el anteproyecto se trató de estimar un cronograma ideal, pero en general cada tarea resulta ser más compleja de lo previsto. Algunos cambios aparentemente triviales pueden llevar días de trabajo, mientras que otros que en un principio resultan imponentes, pueden acabar solucionándose en pocas horas. No obstante, en todo momento se ha tratado de mantener la ambición del proyecto bajo control, centrando la atención siempre en los objetivos principales que se marcaron inicialmente.

Cabe destacar que desde el inicio del desarrollo se han realizado reuniones semanales con la tutora Martina Eckert, el desarrollador del servidor web Miguel Jiménez Arévalo y los estudiantes a cargo de otros módulos en desarrollo, Pablo Lucas Olivares, Bogdán Morar y Moheng Li. Gracias a estas reuniones resultaba sencillo visualizar cuál debía ser el siguiente paso en la elaboración del proyecto, manteniendo las prioridades claras en todo momento. Además, al

intercambiar información y experiencias entre distintos puntos de vista, se facilitó el refinamiento de ciertos aspectos del diseño o la corrección de *bugs* que habrían pasado desapercibidos de no ser así.

También es importante la planificación, no sólo a nivel temporal, sino también estructural. Los primeros scripts que se escriben constituyen los cimientos de la lógica interna del juego. En consecuencia, las decisiones que se toman en las primeras semanas de desarrollo en muchos casos determinan la facilidad o dificultad con la que se pueden añadir nuevas funcionalidades en el futuro. Por suerte, no han surgido grandes problemas en este aspecto durante el desarrollo del juego. No obstante, se nota una tendencia de programación más eficiente y modular conforme se gana familiaridad con el entorno de trabajo.

La creación de este módulo ha supuesto un proceso de aprendizaje constante. Esto se debe principalmente a que los videojuegos son un medio intrínsecamente multidisciplinar. Para llevar a cabo el proyecto ha sido necesario modelar objetos 3D, animarlos, programar numerosos scripts, diseñar la jugabilidad, editar audio e imágenes, o escribir un guion, entre otras tareas. Si bien esto añade complejidad al proyecto, también hace que sea altamente gratificante a nivel creativo.

7. Trabajo futuro

Algo esencial en todo proyecto en el que se desarrolla una herramienta de *software* es la documentación. Por ello, como trabajo futuro se elaborará un GDD (*Game Design Document*) detallando el funcionamiento del juego. La idea es facilitar el trabajo a futuros desarrolladores que vayan a formar parte del equipo de "El Planeta Sonoa". Por tanto, este documento pondrá especial énfasis en explicar el funcionamiento de los *scripts* más complejos del juego.

Existen una serie de caminos que se podrían tomar para ampliar este módulo en concreto. En primer lugar, como se detalló en el apartado 2.1 Desorden del Procesamiento Auditivo Central, la discriminación auditiva consiste en la diferenciación de sonidos con distinta duración, frecuencia y nivel. Por tanto, podría implementarse una tercera cobra que emitiese patrones de notas de nivel alto y bajo. Este modo de juego se comentó con una de las terapeutas de Aurea, pero no se llegó a implementar por falta de tiempo.

En segundo lugar, podría aumentarse la dificultad de las pruebas existentes reduciendo la diferenciación entre distintos tipos de notas. Actualmente las notas cortas y largas tienen una duración fija, al igual que la frecuencia de las notas graves y agudas. Si, por ejemplo, se acorta la duración de las notas largas, o se aumenta la frecuencia de las notas graves, la complejidad de la prueba de discriminación auditiva aumentaría considerablemente. También se podría optar por la opción contraria, y enfatizar aún más la diferencia entre tipos opuestos de notas para hacer el juego más accesible a personas que lo encuentren excesivamente dificultoso.

Otro aspecto se presta a ser ampliado es la obtención de distintos instrumentos mediante el sistema de gemas. Actualmente se pueden obtener tres instrumentos además del *pungi* de madera con el que se empieza. Para ello son necesarias 161 gemas en total. Esta cifra puede parecer alta, pero para los usuarios que jueguen regularmente no será difícil alcanzarla. Se podrían introducir nuevos instrumentos, con distintas condiciones de obtención y precios más altos para que siempre haya una meta hacia la que trabajar y no se pierda la motivación del jugador.

Para concluir, dada la estructura modular de "El Planeta Sonoa", siempre existe la posibilidad de que se añada un nuevo continente. Actualmente, se considera que los seis continentes que conforman el juego cubren los ejercicios de entrenamiento auditivo más solicitados por los terapeutas. Por tanto, no se planea expandir el juego de forma inmediata. No obstante, una vez el juego esté disponible en un mayor número de centros audiológicos podría darse el caso de que los terapeutas soliciten un nuevo tipo de ejercicio auditivo. Si esto ocurriera, la estructura del proyecto facilitaría la integración de un séptimo continente.

8. Bibliografía

[1] B. Nicoll y B. Keogh, de *The Unity Game Engine and the Circuits of Cultural Software*, Palgrave Pivot, Cham, 2019, pp. 1-21.

[2] P. Castro y J. Yadira, « Blexer. Desarrollo de videojuego terapéutico para personas con movilidad reducida.,» Madrid, 2018.

[3] SoundStorm, «SoundStorm,» [En línea]. Available: https://www.soundstorm.app/. [Último acceso: 10 Julio 2023].

[4] Carnegie Learning, «Fast ForWord | Carniege Learning,» [En línea]. Available: https://www.carnegielearning.com/solutions/literacy-ela/fast-forword/. [Último acceso: 2023 07 10].

[5] T. J. Bellis, «Deficit-Specific Intervention for Auditory Processing Disorders,» de *Assesment and Management of Central Auditory Processing Disorders in the Educational Setting: From Science to Practice*, San Diego, Plural Publishing, 2011, pp. 352-354.

[6] F. E. Musiek, J. A. Baran y E. Schochat, «Selected management approaches to central auditory processing disorders,» *Scand Audiol,* vol. 28, nº 51, pp. 63-76, 1999.

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

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

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

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

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

[12] P. L. Olivares, «Ampliación de un videojuego terapéutico para mejora del procesamiento auditivo: creación de un módulo de escucha dicótica.,» Madrid, 2023.

[13] 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.

[14] Gobierno de India, «Pungi | Indian Culture,» [En línea]. Available: https://indianculture.gov.in/node/2686867. [Último acceso: 10 Julio 2023].

[15] Blender Default Cube, «Low Poly Cobra Snake modeling in Blender - Youtube,» [En línea]. Available: https://www.youtube.com/watch?v=GhMNCUV2V0U. [Último acceso: 10 Julio 2023].

[16] Blender Default Cube, «Vulture Modeling in Blender 2.93 - Youtube,» [En línea]. Available: Vulture Modeling in Blender 2.93. [Último acceso: 10 Julio 2023].

[17] Solvkart, «Modelo 3D de cofre estilizado gratuito - TurboSquid 1597758,» 28 Julio 2020. [En línea]. Available: https://www.turbosquid.com/es/3d-models/stylized-chest-animations-3d-model-1597758#. [Último acceso: 10 Julio 2023].

[18] 23 Space Robots and Counting..., «Free Low Poly Desert Pack | 3D Environments | Unity Asset Store,» 9 Enero 2018. [En línea]. Available: https://assetstore.unity.com/packages/3d/environments/free-low-poly-desert-pack-106709. [Último acceso: 10 Julio 2023].

[19] SketchUp Texture Club, «Beach sand texture seamless 12758,» [En línea]. Available: https://www.sketchuptextureclub.com/textures/nature-elements/sand/beach-sand-texture-seamless-12758. [Último acceso: 10 Julio 2023].

[20] L. Molinero, «Free photo | Sand ground textured.,» [En línea]. Available: https://www.freepik.com/free-photo/sand-ground-textured_1198441.htm. [Último acceso: 10 Julio 2023].

[21] 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].

[22] Makalina, «Stylized Cartoon Medieval market set Free low-poly 3D model | CGTrader,» [En línea]. Available: https://www.cgtrader.com/free-3d-models/exterior/historic-exterior/medieval-market-set. [Último acceso: 10 Julio 2023].

[23] Solumn Night, «Low Poly Pack - Environment Lite | 3D Exterior | Unity Asset Store,» 19 Octubre 2017. [En línea]. Available: https://assetstore.unity.com/packages/3d/props/exterior/low-poly-pack-environment-lite-102039. [Último acceso: 10 Julio 2023].

[24] mdubreu823564 , «View of a desert In Dubai 1271913 Stock Photo At Vecteezy,» [En línea]. Available: https://www.vecteezy.com/photo/1271913-view-of-a-desert-in-dubai. [Último acceso: 2023 Julio 10].

[25] Creepy Cat, «3D Games Effect Pack Free | VFX Particles | Unity Asset Store,» [En línea]. Available: https://assetstore.unity.com/packages/vfx/particles/3d-games-effects-pack-free-42285. [Último acceso: 10 Julio 2023].

[26] Brackeys, «FIRST PERSON MOVEMENT in Unity - FPS Controller - Youtube,» 27 Octubre 2019. [En línea]. Available: https://www.youtube.com/watch?v=_QajrabyTJc. [Último acceso: 10 Julio 2023].

[27] Unity Ace, «First Person Camera in Unity - Youtube,» 7 Febrero 2022. [En línea]. Available: https://www.youtube.com/watch?v=5Rq8A4H6Nzw&t=207s. [Último acceso: 10 Julio 2023].

[28] Hellska, «Freesound - "flute_note_tremolo.wav" by hellska,» 15 Noviembre 2015. [En línea]. Available: https://freesound.org/people/hellska/sounds/328727/. [Último acceso: 10 Julio 2023].

[29] Phrydom, «Freesound - "c-5.wav" by phrydom,» 25 Enero 2006. [En línea]. Available: https://freesound.org/people/phrydom/sounds/14845/. [Último acceso: 10 Julio 2023].

[30] Sandyrb, «Freesound - "BRASS END IN A 001.wav" by sandyrb,» 17 Diciembre 2009. [En línea]. Available: https://freesound.org/people/sandyrb/sounds/85892/. [Último acceso: 10 Julio 2023].

[31] Native Instruments, «Komplete Start: Free virtual instrument, sounds, & effects bundle,» [En línea]. Available: https://www.native-instruments.com/es/products/komplete/bundles/komplete-start/. [Último acceso: 10 Julio 2023].

[32] AlterEgo, «AlterEgo Voice banks,» [En línea]. Available: https://www.plogue.com/products/voice-banks.html. [Último acceso: 10 Julio 2023].

[33] Reaper, «Reaper | Audio Production Without Limits,» 2023. [En línea]. Available: https://www.reaper.fm/. [Último acceso: 10 Julio 2023].

[34] A. Dellinger, «Gaming's Environmental Impact is Bigger Than You Think,» Plogue, 29 Enero 2020. [En línea]. Available: https://www.plogue.com/products/voice-banks.html. [Último acceso: 10 Julio 2023].

Anexo I: Presupuesto

En esta sección se pretende determinar el coste del proyecto, teniendo en cuenta tanto las horas de trabajo invertidas como el valor monetario de los dispositivos empleados para su desarrollo. El ordenador empleado para el desarrollo del videojuego ha sido un portátil Asus Zenbook 14 UX431FL-AM049T, valorado en 999€. El dispositivo consta de las siguientes especificaciones técnicas:

- Procesador: Intel CoreTM i7-10510U (4 Núcleos, 8 Subprocesos, Caché: 9 MB SmartCache, 1.80 GHz hasta 4.90 GHz, 64-bit).
- Memoria RAM: 16 GB LPDDR3 2133 MHz.
- Almacenamiento: 512 GB SSD M.2 PCIEG3x2 NVME.
- Controlador gráfico: NVIDIA GeForce MX250 2GB GDDR5 VRAM.
- Sistema operativo: Windows 10 Home (64Bits).

El ordenador se ha usado en conjunto con periféricos, como un ratón Logitech M185 (14,00 €) o unos auriculares Sony MDR-7506 (105,00 €). Para grabar las voces de los personajes se ha empleado un micrófono de condensador RODE NT1-A (169,00 €) y una interfaz de audio Steinberg UR22 (120,00 €).

En cuanto al software empleado, Unity, Visual Studio, Blender, GIMP, Audacity, Reaper, AlterEgo son programas que o bien son gratuitos, u ofrecen un periodo de prueba indefinido como es el caso de Reaper. Sin embargo, parte de la edición de audio y diseño de sonido se ha realizado en Ableton Live 11 Suite, cuya licencia cuesta 599,00 €. No obstante, se podría haber realizado perfectamente en la versión “Intro” de Ableton, cuya licencia es más económica, costando 79,00 €.

En cuanto a la mano de obra, han sido necesarias unas 400 horas de trabajo por parte de un ingeniero de telecomunicaciones junior. El salario anual bruto de un ingeniero de telecomunicaciones recién graduado es de unos 20.600€ [1]. Si se considera una jornada completa de 160 horas mensuales, cada hora de trabajo tendría un coste de 10,73 € aproximadamente. Esto resultaría en 4292,00 € por el trabajo realizado. La tabla 2 desglosa todos los costes del proyecto y presenta la cifra del presupuesto total.

Tabla 2. Cálculo del presupuesto del proyecto.

Descripción	Unidades	Coste unitario	Coste por horas	Horas	Total
Material					
Ordenador	1	999,00 €	-		999,00 €
Ratón	1	14,00 €	-		14,00 €
Auriculares	1	105,00 €	-		105,00 €
Micrófono	1	169,00 €	-		169,00 €
Interfaz de audio	1	120,00 €	-		120,00 €
			Parcial		1407,00 €
Licencias de software					
Ableton Live Intro	1	79,00 €	-		79,00 €
Resto de programas	1	0,00 €	-		0,00 €
			Parcial		79,00 €
Personal					
Ingeniero junior	1	-	10,73€	400	4292,00 €
			Parcial		4292,00 €
Total					**5778,00 €**

Por tanto, se estima un presupuesto total de 5778,00 €. Si bien es cierto que se podrían recortar algunos gastos utilizando dispositivos más baratos y limitándose a usar software gratuito, el grueso del presupuesto se invierte en la mano de obra. En consecuencia, el resultado final no variaría excesivamente. Por otro lado, el sueldo del personal es más bien bajo, ya que se ha tomado como referencia el salario de un ingeniero recién egresado. En conclusión, la estimación se puede considerar realista.

Referencias del Anexo I: Presupuesto

[1] Jobted, «¿Cuánto Cobra un Ingeniero de Telecomunicaciones (Sueldo 2023) | Jobted.es,» 2023. [En línea]. Available: https://www.jobted.es/salario/ingeniero-telecomunicaciones [Último acceso: 10 de Julio de 2023].

Anexo II: Manual

En esta sección se explica cómo interactuar con el juego. En primer lugar, desde el punto de vista de un usuario, y, posteriormente, se incluye una sección dedicada a la interfaz web para los terapeutas.

Guía para el usuario

1. Menú de configuración local

Al iniciar el juego, se pedirá al jugador su nombre de usuario y contraseña. Si se decide continuar sin usuario y se selecciona el continente Kobarag, aparecerá en pantalla el menú interno del juego (figura 22). Se puede interactuar con este menú usando el ratón para configurar los ajustes de la partida. Estos incluyen varios parámetros para ajustar la dificultad, controles, tipo de sonidos y mezclador de audio.

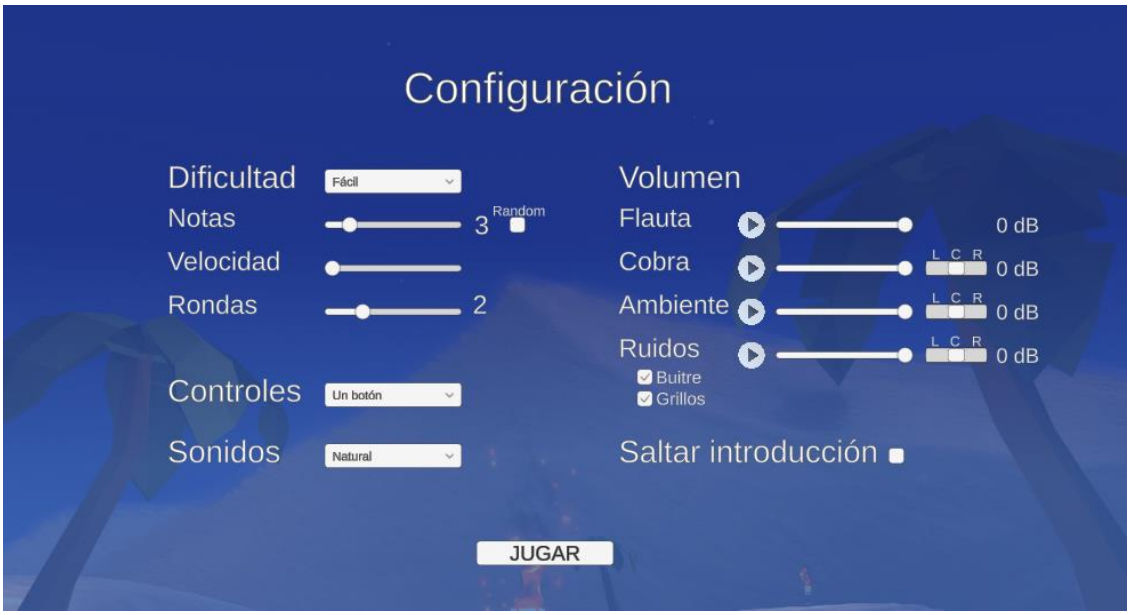

Figura 22. Menú de configuración en modo local.

Al pulsar el botón "JUGAR", se inicia el juego y comienza a reproducirse la cinemática introductoria. Puede saltarse pulsado 1 (botón A en el mando de Xbox). El jugador aparece frente al mercado.

2. Control de movimiento y cámara

Los controles de movimiento son los siguientes:

- Tecla W o ↑ para avanzar hacia delante.
- Tecla S o ↓ para avanzar hacia atrás.
- Tecla A o ← para avanzar hacia la izquierda.
- Tecla D o → para avanzar hacia la derecha.

La cámara se puede rotar moviendo el ratón. Si se usa un mando, el movimiento está asignado al joystick izquierdo y el control de cámara está asignado al joystick derecho. En la figura 23 se aprecia un diagrama con los controles mencionados.

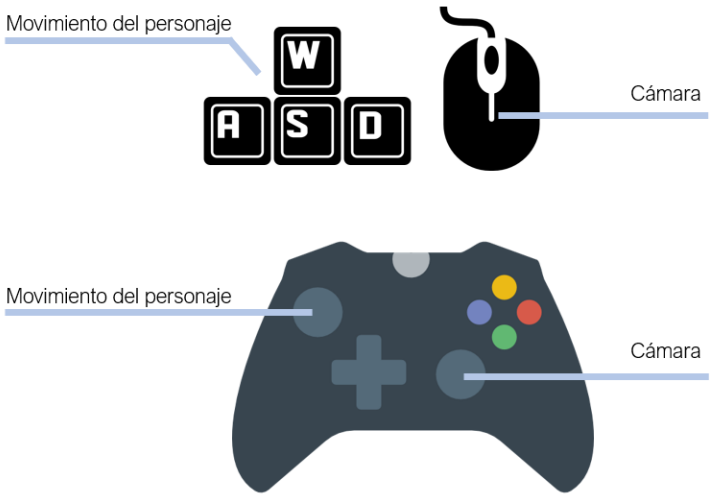

Figura 23. Control de movimiento y cámara usando teclado y ratón (arriba) y mando de Xbox (abajo).

3. Menú de pausa

Se puede acceder al menú de pausa (figura 24) pulsando la tecla Esc. Haciendo clic en "Continuar", se vuelve al juego, mientras que haciendo clic en "Salir" se vuelve al menú inicial (el planeta de selección de continente).

Figura 24. Menú de pausa de Kobarag.

4. Iniciar prueba de memoria auditiva

Para iniciar una ronda, el jugador debe acercarse a alguna de las dos cobras centrales y mirarlas. A la izquierda se encuentra Dora, la cobra verde que canta melodías de notas largas y cortas. A la derecha se encuentra Antón, la cobra morada que canta melodías de notas graves y agudas. La cobra preguntará al jugador si quiere empezar a jugar. Se puede pulsar la tecla 1 (botón A en mando de Xbox) para aceptar, o la tecla 2 (botón B en mando de Xbox) para rechazar la propuesta.

5. Modo dos botones

- Tecla 1 para introducir una nota corta o grave. (Botón A si se usa mando de Xbox).
- Tecla 2 para introducir una nota larga o aguda. (Botón B si se usa mando de Xbox).
- Tecla R para solicitar que se repita la secuencia de notas. (Botón X si se usa mando de Xbox).
- Tecla E para cambiar de *pungi* (si se ha obtenido alguno). (Botón Y si se usa mando de Xbox).

En la figura 25 se muestra un diagrama del modo dos botones.

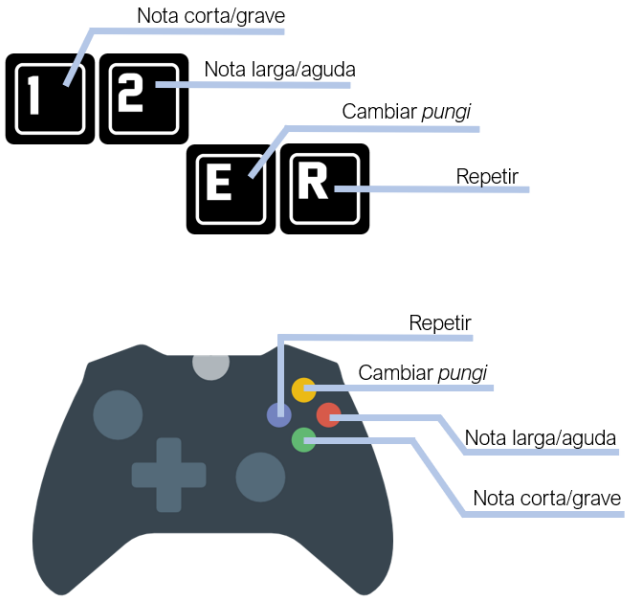

Figura 25. Modo dos botones con teclado (arriba) y mando de Xbox (abajo).

6. Modo un botón

Este modo de juego sólo es aplicable a la prueba de notas cortas y largas (cobra verde).

- Tecla 1 para introducir una nota. Pulsar brevemente para una nota corta, o durante un tiempo para una nota larga. (Botón A si se usa mando de Xbox).
- Tecla R para solicitar que se repita la secuencia de notas. (Botón X si se usa mando de Xbox).

- Tecla E para cambiar de *pungi* (si se ha obtenido alguno). (Botón Y si se usa mando de Xbox).

En la figura 26 se muestra un diagrama del modo un botón.

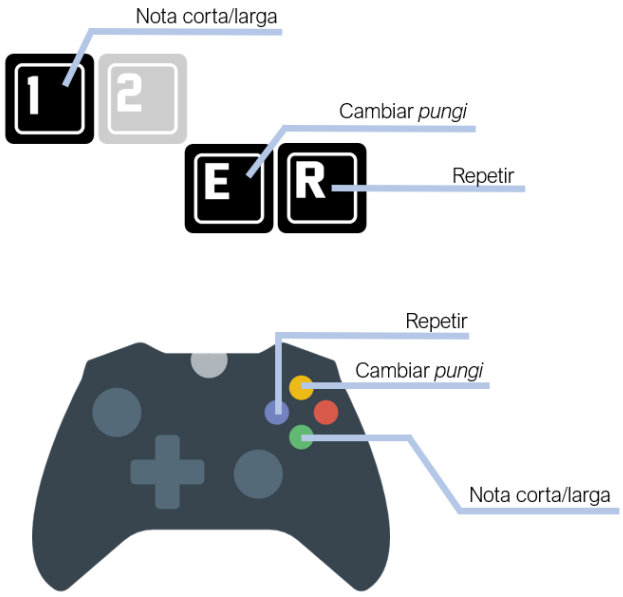

Figura 26. Modo un botón con teclado (arriba) y mando de Xbox (abajo).

Al finalizar todos los niveles se reproduce la cinemática de celebración, que puede saltarse pulsando 1 (botón A en el mando de Xbox).

7. Obtención de *pungis.*

Existen tres instrumentos que se pueden obtener en el juego. Cada uno tiene unas condiciones y método de obtención diferentes. Sin embargo, en todos los casos, es necesario interactuar con objetos o personajes. Para interactuar con Patel, el pozo, el cofre o Camilo se debe pulsar la tecla 1 o el botón A en el mando de Xbox. El jugador debe encontrarse cerca de aquello con lo que quiere interactuar, y cumplir las condiciones de obtención que se indican a continuación:

- *Pungi* cactus. En primer lugar, el jugador debe acercarse al pozo y lanzar una gema. Sólo se puede interactuar con el pozo si se han obtenido gemas previamente. Tras lanzar la gema aparecerá en pantalla el mensaje “Puede que tu deseo se cumpla…”. En ese momento, un cofre aparece al otro lado del oasis, en la esquina opuesta del mapa. Se debe atravesar o rodear el oasis para llegar al cofre y abrirlo.
- *Pungi* dorado. Este instrumento lo vende Patel en su puesto a cambio de 60 gemas y no requiere ningún requisito previo.
- *Pungi* diamante. Este *pungi* lo vende Camilo el camello a cambio de 100 gemas, pero únicamente tras haber obtenido el *pungi* dorado y el *pungi* cactus. Si no se han conseguido lo *pungis* anteriores, no ofrecerá este instrumento al jugador.

La figura 27 muestra la ubicación de cada uno de estos instrumentos en el mapa de Kobarag.

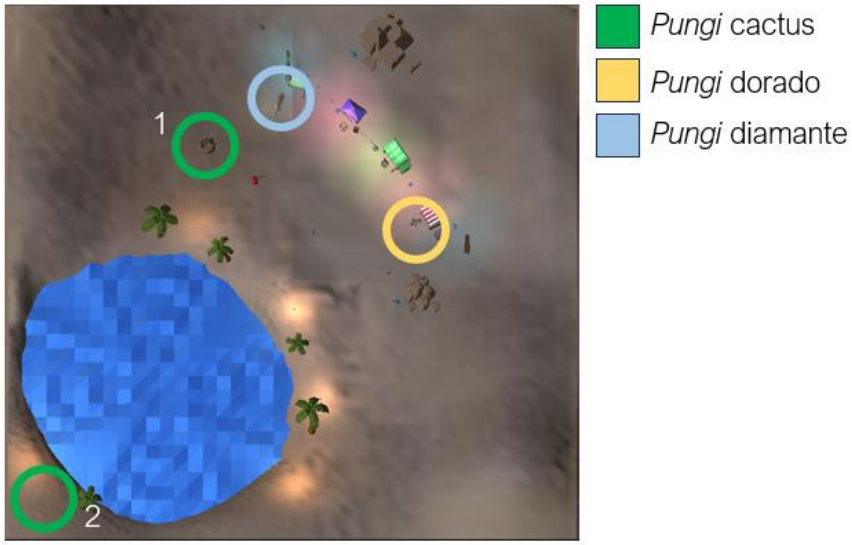

Figura 27. Ubicación de los pungis desbloqueables.

Guía para el terapeuta

1. Añadir una nueva configuración

En primer lugar, se accede a la web http://sonoa.citsem.upm.es y se inicia sesión con el nombre de usuario y contraseña del terapeuta. Se escoge el paciente cuyos ajustes se quieren configurar y a continuación se hace clic en "Kobarag". En pantalla aparecerá el menú realizar una nueva configuración (figura 28). Se aprecia que los ajustes relativos al mezclador de audio son comunes a todos los niveles. Los ajustes de número de rondas, notas velocidad de reproducción o tipo de sonido se pueden configurar individualmente para cada uno de los cinco niveles.

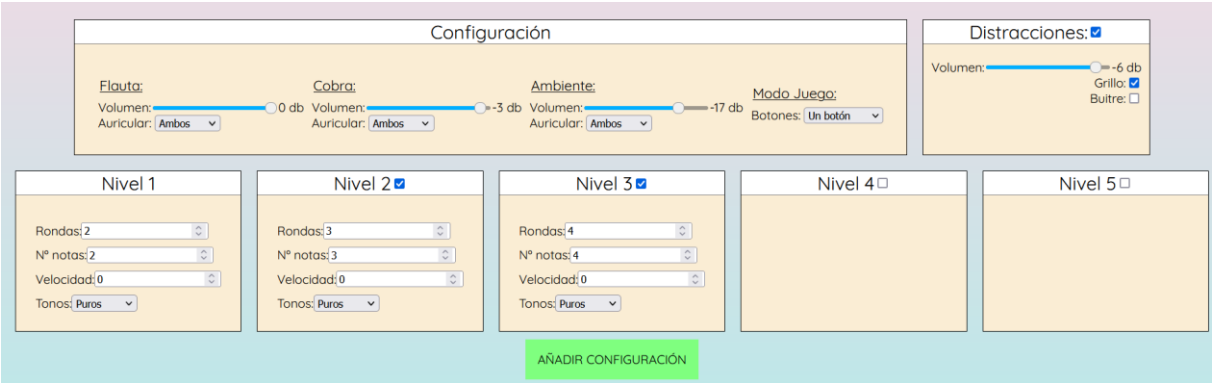

Figura 28. Creación de una nueva configuración en la interfaz web.

Tras asignar los valores deseados a cada parámetro, se hace clic en el botón "Añadir configuración" bajo el menú. Aparecerá una ventana en la que se pide asignar un nombre a la configuración establecida. En este caso se ha elegido el nombre "Ejemplo" (figura 29).

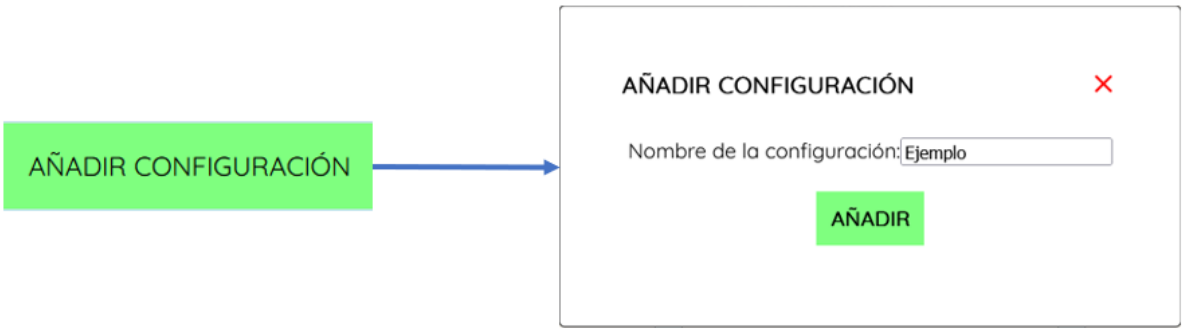

Figura 29. Ventana para añadir configuración.

Una vez añadida la nueva configuración, aparecerá debajo del botón de la figura 29 junto al resto de configuraciones guardadas. Como se aprecia en la figura 30 Las configuraciones se pueden editar mediante el icono del lápiz azul, eliminar pulsando el icono de la papelera roja o revisar mediante el icono del ojo. La configuración que está actualmente seleccionada aparece de color verde. Estos serán los ajustes que se apliquen al juego cuando el usuario inicie sesión. Para cambiar de configuración, se ha de activar la casilla de la esquina inferior derecha.

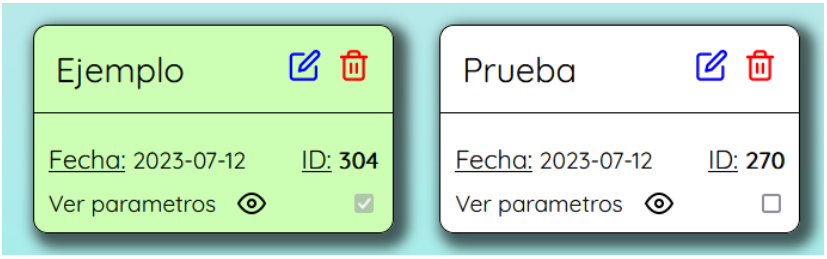

Figura 30. Configuraciones guardadas.

2. Visualización de los resultados

Para acceder a los registros de las sesiones de juego anteriores del paciente, se debe hacer clic en el botón “RESULTADOS” de la barra superior de la interfaz web (figura 31).

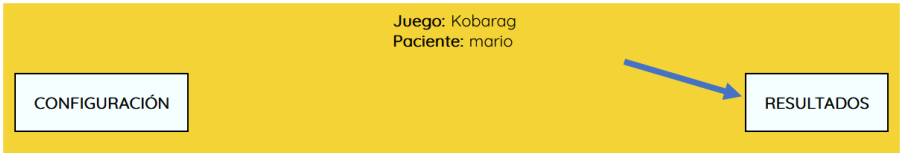

Figura 31. Barra superior de la interfaz web.

Los resultados aparecerán en forma de tabla, pudiendo consultar la hora de inicio y finalización de la ronda, la secuencia de notas generada por el juego y la introducida por el usuario, la velocidad de reproducción, el número de intentos fallidos y repeticiones solicitadas, y por último, los modos de juego y control.

Anexo III: Guion

En este anexo se compilan todos los diálogos del juego, indicando el personaje que dice cada frase y la situación en la que se dice.

En primer lugar, se presentan las líneas de diálogo de Dora y Antón, las cobras principales del juego con las que se realizan ejercicios de memoria auditiva. Al tener guiones prácticamente idénticos se han agrupado las oraciones que para evitar redundancias.

El jugador se acerca a una de las cobras principales.

DORA

¿Te gustaría tocar estas melodías de notas largas y cortas?

ANTÓN

¿Te gustaría tocar estas melodías de notas graves y agudas?

Se acierta una secuencia de notas.

DORA / ANTÓN

¡Eso es! / ¡Bien hecho! / ¡Perfecto!

Se falla una secuencia de notas (primera y segunda vez).

DORA / ANTÓN

Prueba otra vez.

Se cambia la secuencia de notas tras fallar por tercera vez.

DORA / ANTÓN

¿Y esta otra?

Se completa el nivel sin un solo fallo.

DORA / ANTÓN

¡Música para mis oídos!

Se completa el nivel con un solo fallo.

DORA / ANTÓN

¡Eso suena bien!

Se completa el nivel con dos o más fallos.

DORA / ANTÓN

Sigue practicando…

Tras acabar todos los niveles.

DORA / ANTÓN

¿Te gustaría jugar otra vez?

A continuación, se muestran los diálogos de Camilo el camello. Es posiblemente el personaje con más texto ya que presenta los objetivos e historia del juego en la cinemática introductoria.

Cinemática introductoria.

CAMILO

¡Pero qué ven mis ojos! ¡Al fin has llegado! Te doy la bienvenida al gran desierto de Kobarag. Necesitábamos tu ayuda ¡Acompáñame!

Este oasis es el centro de comercio más importante de la región, pero últimamente las cosas no van bien. Dos de las cobras que regentan el mercado están tristes porque llevan mucho tiempo sin oír sus melodías favoritas. A Dora, la cobra verde, le gustan las melodías de notas largas y cortas mientras que, a Antón, la cobra morada, le gustan las melodías de notas graves y agudas. Estas cobras comerciantes son muy generosas y puede que te den gemas que podrás canjear en el puesto de Patel. ¡Mucha suerte!

Frase inicial.

CAMILO

¡Por favor, ayuda a las cobras! Yo no puedo tocar un instrumento con mis pezuñas.

Tras desbloquear el pungi dorado y el pungi cactus.

CAMILO

¡Gracias por devolver la música al desierto! A cambio de 100 gemas te venderé el pungi definitivo.

Si intentas comprar el pungi diamante con menos de 100 gemas.

CAMILO

Vaya… parece que no tiene suficientes gemas.

Tras comprar el pungi diamante.

CAMILO

¡Enhorabuena! ¡El pungi diamante es un instrumento legendario!

Por último, se muestra el guion de Patel, la cobra tendera.

Frase inicial.

PATEL

A mí no me interesa la música... te vendo un pungi especial por 60 gemas ¿qué te parece?

Cuando el jugador intenta comprarlo, pero le faltan gemas.

PATEL

Vuelve cuando tengas suficientes gemas… ¡este no es un pungi cualquiera!

Cuando el jugador lo compra.

PATEL

¡Gracias por tu compra! Usa el pungi dorado para hacer el bien.

Si te vuelves a acercar una vez has comprado el pungi dorado.

PATEL

Cuenta la leyenda que hay un tesoro al otro lado del oasis, pero yo nunca lo he visto…