ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
[GRADUADO EN INGENIERÍA DE SOFTWARE]

[Videojuego imaginativo para el desarrollo de las funciones ejecutivas en la niñez]

[Imaginative videogame for the development of executive functions in childhood]

Realizado por

[Álvaro Nieto González]

Tutorizado por

[Beatriz Barros Blanco]

Departamento

[Lenguajes y ciencias de la computación]

UNIVERSIDAD DE MÁLAGA
MÁLAGA, JUNIO DE 2021

Fecha defensa: por definir

Resumen

Las funciones ejecutivas son un conjunto de procesos mentales complejos utilizados para realizar tareas que nos ayudan a alcanzar metas. Estas funciones progresan durante la niñez y son muy importantes para su correcto desarrollo cognitivo. Se pueden mejorar y desarrollar con aprendizaje y entrenamiento.

Es el presente Trabajo de Fin de Grado (TFG), se realiza una propuesta para entrenar las las funciones ejecutivas de los niños a partir de un videojuego, compuesto de minijuegos enfocados hacia una función específica.

Primeramente, se justifica el contexto del trabajo y se realiza una investigación sobre las funciones ejecutivas y la importancia del juego en el desarrollo de los niños.

A continuación, se ha trabajado en el desarrollo de minijuegos que faciliten el trabajo del docente con las funciones ejecutivas, así como la creación de un hilo conductor, para fomentar el uso de la imaginación de los infantes.

En paralelo, se ha seguido el desarrollo utilizando una metodología ágil y siguiendo las fases propias de la Ingeniería de Software como la toma de requisitos, el modelado del sistema, el desarrollo y la fase de pruebas.

Finalmente, se ha realizado la adaptación de los juegos y su posterior implementación en el portal Siette, incluyendo una parametrización de los minijuegos para ofrecer más opciones y dar más versatilidad a los juegos como herramienta de entretenimiento y de aprendizaje.

Palabras clave:

Funciones ejecutivas, Siette, juego imaginativo, minijuegos, Unity, JavaScript

Abstract

Executive functions are a set of complex brain processes that help us to perform tasks with a goal. These functions progress during childhood and are very important for proper cognitive development

In this End of Grade Work a proposal is realized for the imaginative and executive functions development throught a video game

Firstly, the context of the work is justified, and it is done a research about executive functions and the importance of games in children growth

Next, it has been worked on the development of minigames that help the work of the teachers with de executive functions, as well as the creation of a common thread to promote the childs' imagination use

In parallel, the software has been developed following an agile methodology and following the phases of Software engineering, as requirements gathering or system modeling.

Finalmente, se ha realizado la adaptación de los juegos y su posterior implementación en el portal Siette, incluyendo una parametrización de los minijuegos para hacerlos más personalizables hacia los niños.

Finally, it has been done the adaptation of the games and the integration on Siette's website, including the possibility to customize the minigames through parameters.

Keywords:

Executive functions, Siette, imaginative game, minigames, Unity, JavaScript,

A mis padres y a mi hermana por apoyarme durante toda mi etapa académica

A Ricardo Conejo por su ayuda con la integración

Índice

1

Introducción

1.1 Funciones ejecutivas y juego imaginativo. Proyecto TECHCAT

Las funciones ejecutivas son el conjunto de procesos necesarios para la realización de tareas complejas dirigidas a un objetivo (Arán, 2011), y que facilitan la adaptación del individuo a situaciones nuevas y complejas (Rosselli, Jurado y Matute, 2008).

Estas funciones se desarrollan a muy corta edad, durante los primeros años de vida, un niño comienza a ser capaz de centrarse en un objetivo, e ignorar otros factores que puedan molestar a alcanzarlo. También comienza a ser capaz de mantener y usar la información que le llega.

A partir de los 4 años adquieren un control inhibitorio mucho mayor, pudiendo fijarse metas mucho más ambiciosas con una mejor planificación. Esto puede notarse sobre todo en el lenguaje, donde comienzan a componer frases más complejas con mayor facilidad y rapidez.

Por otro lado, el juego imaginativo es aquel que permite a los niños crear sus propios mundos o hacer juegos de roles, en los que, de alguna forma, se preparan para la vida adulta. El “juego simbólico”, establece las bases para asimilar la complejidad del mundo adulto (Jean Piaget, 1962).

Son múltiples los beneficios estudiados a lo largo de los años de este tipo de juegos, entre ellos se encuentran los siguientes:

- ➢ Practicar la facultad de pensar desde el punto de vista de otra persona.
- ➢ Fomentar el pensamiento narrativo de los niños.
- ➢ Favorece a expresar sentimientos y emociones, así como controlarlas

Los primeros momentos en los que los infantes comienzan a utilizar su imaginación para jugar se da a muy corta edad, llegando a recrear situaciones ya vividas o a imaginar objetos.

A partir de los cuatro años desarrollan considerablemente su imaginación, llegando a simular situaciones sin ayuda de objetos físicos o con objetos menos realistas.

El proyecto que se va a desarrollar es un videojuego tipo aventura gráfica, en el que se dará un escenario en el que el niño pueda situarse. A partir de la toma de decisiones el niño podrá ser parte de la historia contada. Durante el desarrollo del juego el niño comenzará a encontrarse con pequeños desafíos para seguir avanzando con el juego, estos minijuegos continuarán el hijo conductor además de tratar funciones ejecutivas específicas.

Este trabajo está enmarcado dentro del proyecto TECHCAT (Technology Enhance CHildren Cognitive Assessment and Training) (Figura 1). Es un proyecto impulsado desde la universidad de Málaga que busca soluciones tecnológicas que ayuden a observar, diagnosticar y mejorar los procesos asociados con el enriquecimiento cognitivo de los niños, de esta forma integrando las Tecnologías de la información, Neuropsicología y Educación

FIGURA 1 LOGO DEL PROYECTO TECHCAT

1.2 Juegos, juegos educativos y juegos serios

El juego es una actividad de vital importancia en los niños. Además de ser actividades con el fin de entretener o pasar el tiempo, jugar tiene una función fundamental en el desarrollo de los infantes. Entre otras cosas, los niños comienzan a conocer el mundo que les rodea a partir del juego, creando mecanismos para adaptarse a cada situación. También sirve para desarrollar la imaginación, así como la atención, concentración, memoria etc. Es por todo esto por lo que el juego está recogido en el artículo 31 de la Convención sobre los Derechos del Niño, establecido por las Naciones Unidas. Hoy se sabe que el juego no solo promueve la actividad cerebral, sino que también la estructura del cerebro, mejorando el proceso de aprendizaje y las funciones ejecutivas (Yogman, Garner, Hutchinson , HirshPasek , Golinkoff, 2018)

Dentro de los juegos clásicos que todos conocemos, podemos distinguir dos tipos de juegos, los juegos educativos y los juegos serios.

Los juegos educativos están específicamente diseñados con un objetivo de aprendizaje concreto. Debe ser capaz de conseguir que el niño adquiera conocimientos nuevos, sin dejar de lado el interés del niño de obtener entretenimiento y disfrute. Y esto último es la gran ventaja de los juegos educativos, ya que, para los niños, estas actividades lo relacionan con una forma de diversión y pasatiempo, por ello van a recibirlas con atención y van a comprometerse y rendir mucho más ya que para ellos no hay diferencia entre jugar y aprender.

Por otro lado, los juegos serios podemos definirlos como aquellos juegos cuyo objetivo principal no es la diversión o el entretenimiento, sino el aprendizaje o la práctica de habilidades (*Todo sobre serious games y game-based learning*, 2017). A pesar de que incluyan elementos de la gamificación, el objetivo no tiene por qué ser que el usuario pase un rato agradable. Muchos de los ejemplos de juegos serios son los de simulación, donde se busca una gran fidelidad con una práctica real para que el jugador pueda aprender de esa experiencia sin tener que vivirla en el mundo real.

1.3 Objetivos del TFG

El objetivo del TFG es crear una herramienta atractiva para los niños que fomente el enriquecimiento de su imaginación y las funciones ejecutivas. Para lograr esto, se implementará un videojuego que siga una historia que los niños puedan seguir y puedan ser partícipes, obteniendo así los beneficios del juego imaginativo, y que esté compuesto por pequeñas pruebas o minijuegos relacionados con las funciones ejecutivas

Para la realización de esto se han planteado los siguientes subobjetivos:

- Estudiar las herramientas necesarias para la implementación de juegos de ordenador y su portabilidad a dispositivos Tablet.
- Definir los parámetros y los indicadores necesarios para la evaluación del juego de parte de un docente
- Implementar el juego para su funcionamiento en ordenador, así como la implementación de los minijuegos en el portal Siette.

1.4 Plan de trabajo

Para la realización de este proyecto se ha seguido un desarrollo ágil tipo Scrum, donde se han realizado iteraciones de desarrollo en el que al final de las mismas se ha obtenido un producto probable para el director del TFG, consiguiendo de esta forma acercarse a una solución que satisfaga las necesidades tanto de profesores como niños.

Las fases en las que se ha realizado el trabajo son las siguientes:

- ➢ Definición de una historia acorde a un tema atractivo para los niños.
- ➢ Definición y desarrollo de varios minijuegos.
- ➢ Definición de indicadores para la evaluación del niño.
- ➢ Integración de un subconjunto de los minijuegos desarrollados en el entorno Siette,

1. 5 Organización de la memoria

La memoria del proyecto está estructurada de la siguiente forma:

- ➢ Estudio del contexto del proyecto
- ➢ Análisis de las tecnologías y herramientas utilizadas para su implementación.
- ➢ Explicación del juego a desarrollar.

- Desarrollo del proceso de creación.
- Conclusiones.

2

Contexto del trabajo

2.1 Funciones ejecutivas

Como ya he comentado, las funciones ejecutivas son las herramientas que dirigen nuestra conducta, así como nuestra actividad cognitiva y emocional para la consecución de objetivos o metas.

2.2.1 Taxonomía de las funciones ejecutivas

Estas funciones ejecutivas pueden clasificarse de diferentes formas, en la siguiente clasificación (Castillero, s.f.), se encuentran agrupadas según al objetivo que ayuda a conseguir al ser humano.

- Razonamiento: Ser capaz de emplear las distintas informaciones disponibles y ver las conexiones que existen entre ellas.
- Planificación: Nos permite elaborar planes de actuación, es decir, programar una serie de pasos para alcanzar la meta que estamos buscando.
- Fijación de metas: Esta habilidad es la que nos permite decidir sobre cómo vamos a invertir nuestras energías. Está muy vinculada a la motivación.
- Toma de decisiones: Capacidad de escoger una opción entre varias que se nos pueden presentar.

- ➢ Inicio y finalización de tareas: La decisión de cuándo hay que comenzar y cuándo se ha de terminar una tarea requiere de pensamientos muy elaborados e implica una actividad cognitiva muy importante.
- ➢ Organización: La capacidad de reunir, estructurar y secuenciar información de manera eficiente.
- ➢ Inhibición: La habilidad que nos permite regular nuestros actos mediante la detención de la conducta, es decir, esta función es la que hace que seamos capaces de resistir impulsos concretos. Es una de las funciones más importantes y estudiadas.
- ➢ Monitorización: La capacidad de mantener la atención sobre una tarea y saber cómo se está realizando la misma, para poder hacer correcciones en el caso de que surjan problemas imprevistos.
- ➢ Memoria de trabajo verbal y no verbal: Capacidad de almacenar de manera temporal datos y de procesarlos.
- ➢ Anticipación: La habilidad que permite al individuo prever de antemano resultados de una acción y sus consecuencias
- ➢ Flexibilidad: Capacidad que permite cambiar el modo de pensar o de actuar sobre una tarea ante posibles cambios.

2.1.2 Sistemas informáticos para el enriquecimiento de las funciones ejecutivas

Durante los últimos años han surgido muchas empresas relacionadas con la educación buscando desarrollar aplicaciones que sirvieran para fomentar el entrenamiento de las funciones ejecutivas. En la mayoría de los casos, los sistemas implementados han resultado siendo videojuegos, ya que suponen una actividad sobre todo entre los más pequeños, además de que consiguen el objetivo de entrenar las funciones ejecutivas alejándose del ámbito clásico del aprendizaje y suponiendo más una actividad lúdica que un esfuerzo para la persona.

Entre estos sistemas cabe destacar el famoso videojuego *Big Brain Academy* desarrollado por Nintendo. En este juego el jugador debía realizar una serie de pruebas, que suponían un desafío a nivel cognitivo y donde se estimulaban

múltiples funciones ejecutivas en cada una de ellas. Al completar todas las pruebas, aparece un resumen con el rendimiento ofrecido por el jugador. Además de esto, ofrece un modo práctica donde realizar una prueba en concreto, y un modo competición, donde se jugaba de manera online con hasta 7 jugadores más para ver quién conseguía la mejor puntuación.

Además de juegos comerciales, existen múltiples juegos gratuitos disponibles para dispositivos Android y iOS en sus respectivas tiendas. Un ejemplo de esto puede ser *Rush Traffic* un juego en el que el objetivo es poder sacar a todos los coches de un parking, solo permitiendo el movimiento hacia delante o atrás de los mismos. En este juego se practica sobre todo la anticipación y la planificación.

2.2 Juegos para niños

Los niños juegan porque la actividad lúdica le permite ir estructurando y evolucionando en su personalidad (Meneses-Montero & Monge-Alvarado, 2011). Hay distintos tipos de juegos para las diferentes edades. Por ejemplo, existen el juego funcional, que se basa en la manipulación y la exploración del mundo que rodea al infante, dura hasta los seis meses de edad; hasta los dos años, se da el tipo de juego de autoafirmación, donde el niño conquista una mayor habilidad motora que le va a dar confianza en sus propios medios, autonomía e iniciativa.

A partir de los dos, y hasta los cuatro años, se da el juego simbólico, donde predominan los juegos de construcción y destrucción, donde se comienza con la imitación y la simulación de experiencias.

Por último, se da el juego prosocial, que suele durar hasta los seis años. Es aquí donde el niño comienza a buscar compañeros para jugar, aunque a esta edad aún no son capaces de organizar un juego, cada uno se limita a asociarse y representar un papel dentro del mismo

2.2.1 Juego tangible vs videojuego

Como ya he mencionado, el juego va evolucionando junto al crecimiento del niño ya que este tiene otros intereses y metas. La mayoría de los juegos durante edades tempranas corresponden a juegos tangibles, que no son más que aquellos en los que los infantes interactúan con objetos físicos de su alrededor, ya sea una pelota, un coche de juguete, una pieza de un puzle etc.

Generalmente a una edad más avanzada, los niños comienzan a jugar con los videojuegos. El mundo de los videojuegos siempre ha estado vinculado a la polémica tanto dentro del ámbito de la educación como fuera de ella. Aunque han sido ya numerosos casos de estudio que avalan que los videojuegos pueden llegar a ayudar al desarrollo de las capacidades de los niños y al fortalecimiento de sus funciones ejecutivas, aún existe cierto rechazo por parte de algunos docentes a incluirlos como actividad educativa.

Por un lado, los juegos tangibles ofrecen una experiencia más accesible, en la que el niño utiliza más su imaginación para recrear situaciones, o en la que puede cambiar las reglas del juego y adaptarlas a su gusto. También cabe destacar, que los juegos tangibles van muy ligados al favorecimiento de la psicomotricidad del individuo.

Por otro lado, los videojuegos ofrecen una experiencia más compleja, en la que los jugadores deben comprender y adaptarse a unas reglas impuestas, pero que otorgan una experiencia a menudo más realista y fiel al mundo real, permitiendo, por ejemplo, la creación de los previamente mencionados juegos serios.

2.2.2 Juegos para enriquecer las Funciones ejecutivas

Existen gran variedad de juegos para entrenar las funciones ejecutivas de los niños. Entre ellos, los juegos de cartas y juegos de mesa son especialmente buenos para mejorar la función ejecutiva de la memoria de trabajo (Center on the Developing Child. Harvard University s.f.). Juego como el de *Las parejas*, en el que los niños deben ir levantando cartas y buscando su pareja es un claro ejemplo. Otros juegos de mesa que requieren respuestas rápidas como el *Jungle Speed*, en el que se van levantando cartas de un montón y si coincide con alguna de las cartas de tu mano debes coger el palo que se encuentra entre los jugadores antes que los demás, fomenta las funciones ejecutivas de inhibición.

Juegos de estrategia clásicos como el *Parchís* o *Hundir la flota* son una buena práctica para estimular las funciones ejecutivas de la inhibición, memoria de trabajo, planificación y toma de decisiones.

Aunque no es necesario que un juego requiera de únicamente pensamiento lógico para que sea beneficioso para las funciones ejecutivas. Juegos en los que

además se requiere de cierta actividad física pueden tener en cuenta estas funciones, como, por ejemplo, para *El juego de la silla* es necesaria la función ejecutiva de la inhibición, o juegos de imitación, donde es necesaria la memoria de trabajo.

En definitiva, muchos de los juegos clásicos a los que todos hemos dedicado tiempo alguna vez en la vida, sirven como actividad para el enriquecimiento de las funciones ejecutivas, sin perder en ningún momento la actividad lúdica que al fin y al cabo es lo más interesante para los niños.

2.3 Indicadores y evaluación

Los indicadores y la evaluación de los minijuegos son una tarea clave en el proceso, ya que estos son los encargados de poder evaluar el rendimiento de los niños en cada una de las actividades.

Los indicadores se encargan de definir qué parámetros van a evaluarse durante el juego. En juegos en los que se quiere medir funciones ejecutivas como la memoria de trabajo, el número de fallos es uno de los indicadores más importante para medir el desarrollo de la función. Por otro lado, en juegos centrados en la función ejecutiva de toma de decisiones, sería interesante tomar como referencia el número de decisiones correctas tomadas por el jugador, o el tiempo que le lleva tomar estas decisiones.

Algunos de los indicadores que se van a definir son comunes a todo el conjunto de tareas, como puede ser el número de aciertos o de fallos, o el tiempo que se ha tardado en completar el juego. Otras, por el contrario, son específicas de cada juego.

Uno de los indicadores tomados en cuenta en el juego *Sol/Luna*, que se analizará posteriormente, es el tiempo medio de respuesta. Debido a que, uno de los componentes clave para que ese minijuego realmente enriquezca las funciones ejecutivas a las que está ligado es que el tiempo de respuesta al producirse un evento sea lo más rápido posible.

Otro de los indicadores a tener en cuenta para la evaluación será el número de intentos para llegar a la solución. También será utilizado en algunas tareas en

específico en el que para superar el juego será necesario realizar una serie de aciertos de forma consecutiva

2.4 Tecnologías y herramientas

Dentro de la industria de los videojuegos, las empresas desarrollan sus productos utilizando herramientas propiamente desarrolladas por las mismas. Entre este software utilizado por las empresas se encuentra el motor gráfico o el motor físico.

El motor gráfico es el encargado de dibujar los gráficos en la pantalla del dispositivo en el que se vaya a ejecutar el juego (Palazuelos, 2015). El motor gráfico afecta como se ve la luz, las sombras, las texturas etc.

Por otro lado, el motor físico es aquel que busca definir algoritmos que puedan dar una simulación aproximada a ciertos sistemas físicos, como podría ser los reflejos, las colisiones, los fluidos etc.

Aunque, como ya se ha comentado, la mayoría de las empresas de videojuegos utilizan softwares propios para el desarrollo de sus juegos, existen juegos muy populares en el mercado que han sido desarrollado con motores de juegos gratuitos como el usado en este proyecto

2.4.1 Unity

Unity (Figura 2) es un motor de videojuegos multiplataforma lanzado por Unity Technologies en el año 2005. Permite la creación de videojuegos tanto en 2D como en 3D a través de un entorno gráfico sencillo e intuitivo con el que poder ver prácticamente en tiempo real los cambios en el proyecto.

Unity se basa en la creación de componentes o “gameObjects” que no son más que cada uno de los elementos vistos en pantalla. Cada uno de ellos tiene asociado un script, escrito en el lenguaje de programación C#

Otro de los aspectos positivos de Unity es la exportación de proyectos. Permite la exportación a PC, dispositivos móviles, Web e incluso consolas de última generación.

Alguno de los juegos más importantes desarrollados con Unity son Hearthstone, el popular juego de cartas con más de 20 millones de jugadores activos y Pokémon Go que reúne a más de 100 millones de jugadores.

FIGURA 2: LOGO DE UNITY

2.4.2 Javascript

JavaScript (Figura 3) es un lenguaje de programación que permite implementar funciones complejas en la web. Es usado para hacer que las páginas web puedan responder a eventos y que no se limiten a mostrar simplemente contenido estático.

Originalmente desarrollado por Netscape, JavaScript se ha convertido con el paso de los años en la tercera capa de la implementación de sistemas web, junto a HTML y CSS.

Destaca por su simpleza y su capacidad de responder a eventos en tiempo real desde el lado del cliente, así como cambiar elementos de la página o hacer cálculos.

A diferencia de Java, con el que puede llegar a haber confusión, JavaScript no es un lenguaje orientado a objetos, y no necesita ningún compilador sino que será el propio navegador el encargado de interpretar el código.

FIGURA 3: LOGO CLÁSICO DE JAVASCRIPT

2.4.3 Adobe Illustrator

Adobe Illustrator (Figura 4) es un software diseñado para realizar dibujos vectoriales. Los gráficos vectoriales son dibujos basados en puntos unidos mediante líneas rectas, esto tiene como objetivo que la imagen general mantenga la calidad y coherencia sin importar la solución.

Como aspecto positivo destacar su interfaz gráfica y su potencia. La posibilidad de escalar las imágenes sin perder calidad hace de Illustrator una de las herramientas más potentes del mercado.

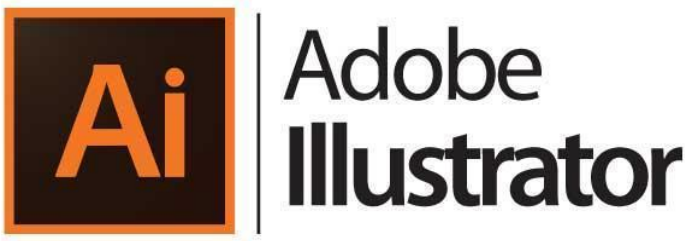

FIGURA 4: LOGO ADOBE ILLUSTRATOR

2.4.4 Siette

SIETTE, de sus siglas Sistema Inteligente de Evaluación mediante Test para TeleEducación (Figura 5), es una herramienta para la aplicación de test y pruebas adaptativas a través de internet.

Ofrece al profesorado y alumnado un marco flexible, sirviendo como entorno para alojar los test creados por los alumnos, y a estos la posibilidad de poder realizarlos. Además de esto, ofrece a los profesores poder crear los test a través

de internet, así como conocer el estudiar la efectividad de los test, calcular los parámetros del conjunto de preguntas o estudiar el funcionamiento de los test en situaciones controladas.

La integración de los minijuegos a la web se va dar a partir de esta aplicación, facilitando el trabajo del profesor tanto en la preparación del juego como en la evaluación del mismo

FIGURA 5: LOGO DE SIETTE

3

Juego imaginativo para entrenar las funciones ejecutivas

La creación de un videojuego imaginativo implica la elaboración de un guion de una historia con la que los niños puedan sentir interés. Debido a la edad de los niños a los que va dirigido el juego, se ha optado por un juego del estilo “juego narrativo”. Los juegos narrativos son aquellos en los que los participantes tienen que crear una historia a partir de determinados elementos, y tienen la capacidad

de fomentar la creatividad y la improvisación (Mallén, 2016). Algunos videojuegos de este estilo consisten en escenas en las que un jugador debe tomar una decisión, algunos ejemplos pueden ser *Beyond: Two souls* o *Heavy Rain* (Figura 6). El videojuego desarrollado busca, aunque a menor escala que los ejemplos mencionados, una idea similar de toma de decisiones donde los niños deberán ponerse en la piel del personaje y tomar lo que consideran la mejor decisión para conseguir el objetivo.

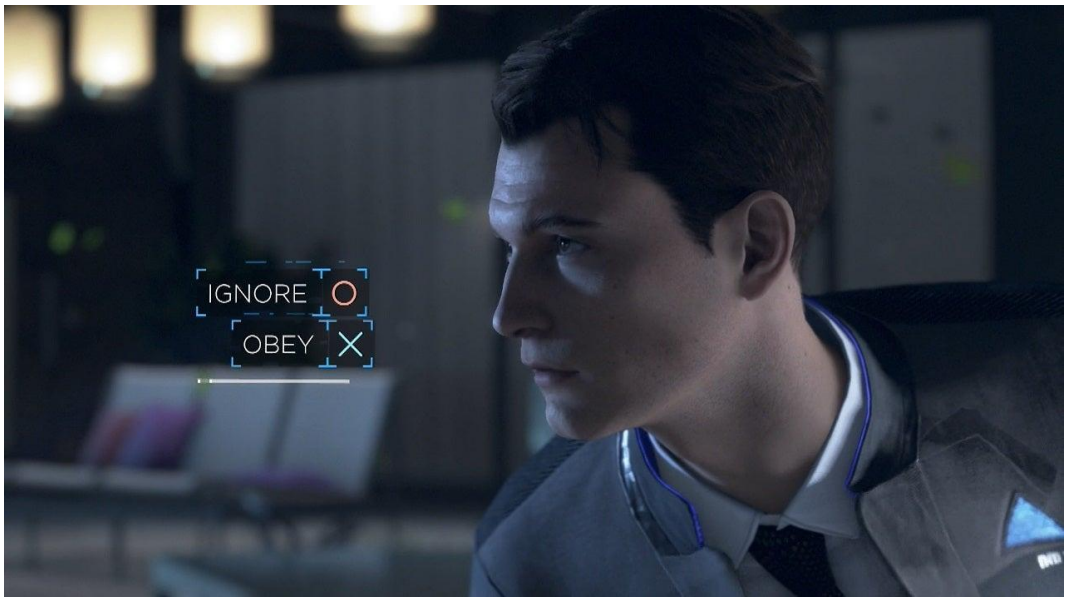

FIGURA 6: DECISIÓN DETNRO DEL JUEGO DETROIT: BECOME HUMAN

A la hora de tomar las decisiones, los seres humanos nos basamos en experiencias personales o las de otros, aunque algunas de ellas son tomadas de forma intuitiva, o incluso las emociones pueden impulsarnos a tomar esa decisión o evitarla.

Son varias las áreas cerebrales las encargadas en tomar decisiones, algunas de ellas están asociadas con la intuición y otras con el razonamiento y el análisis. Cabe destacar que durante la toma de una decisión se estimula la corteza prefrontal, que está altamente relacionada con las funciones ejecutivas, más en concreto con los procesos de planificación, flexibilidad mental y las expectativas sobre los resultados.

A continuación, se detalla el guion seguido en el juego, donde se detallan los diferentes caminos para llegar al final del juego, así como las decisiones que debe tomar el infante para llegar al mismo

3.1 Guion general del juego

El juego desarrollado se basa en un explorador que entra a un templo buscando un tesoro, antes de cada minijuego el jugador tendrá que elegir entre dos opciones. Dependiendo de cada elección aparecerá un minijuego u otro. Las distintas decisiones y minijuegos siguen el diagrama de la Figura 7.

Al comienzo del juego, en la Introducción se presenta al personaje, y se da un breve contexto para que el niño se sitúe y conozca la situación.

La primera elección viene pronto, el jugador deberá elegir entre dos puertas, una de ellas aparentemente cerrada y de la otra puerta sale una ventisca de frío:

- Si elige la puerta cerrada el minijuego consiste en un puzle que se va desarmando con el paso el tiempo
- Si elige la nevada el juego será “Elige el equipaje”, en este minijuego aparecerán ropa y accesorios, el jugador tendrá que elegir los convenientes para ir a la nieve

En la siguiente sala aparecerá un animal perdido en el templo, la siguiente decisión será si ayudar al animal a escapar, o si seguir adelante y no perder tiempo:

- Si elige salvar a los animales el juego consistirá en clasificar a los animales según algunas características
- Si elige seguir adelante entonces tendrá que superar un juego que utiliza el reconocimiento de voz

La siguiente sala tendrá una puerta y un cofre. El personaje le dirá al jugador que ya se ha encontrado el tesoro, es el niño quien elige si continuar a la siguiente sala o si intentar abrir el cofre.

- Si elige el cofre, el juego consistirá en un ejercicio de memorización de parejas con cartas, al final de este minijuego acabará el juego, pero realmente este no era el tesoro que buscaba el explorador.
- Si decide ir por la puerta encontrará el verdadero tesoro, para conseguirlo deberá resolver el juego de Simón dice.

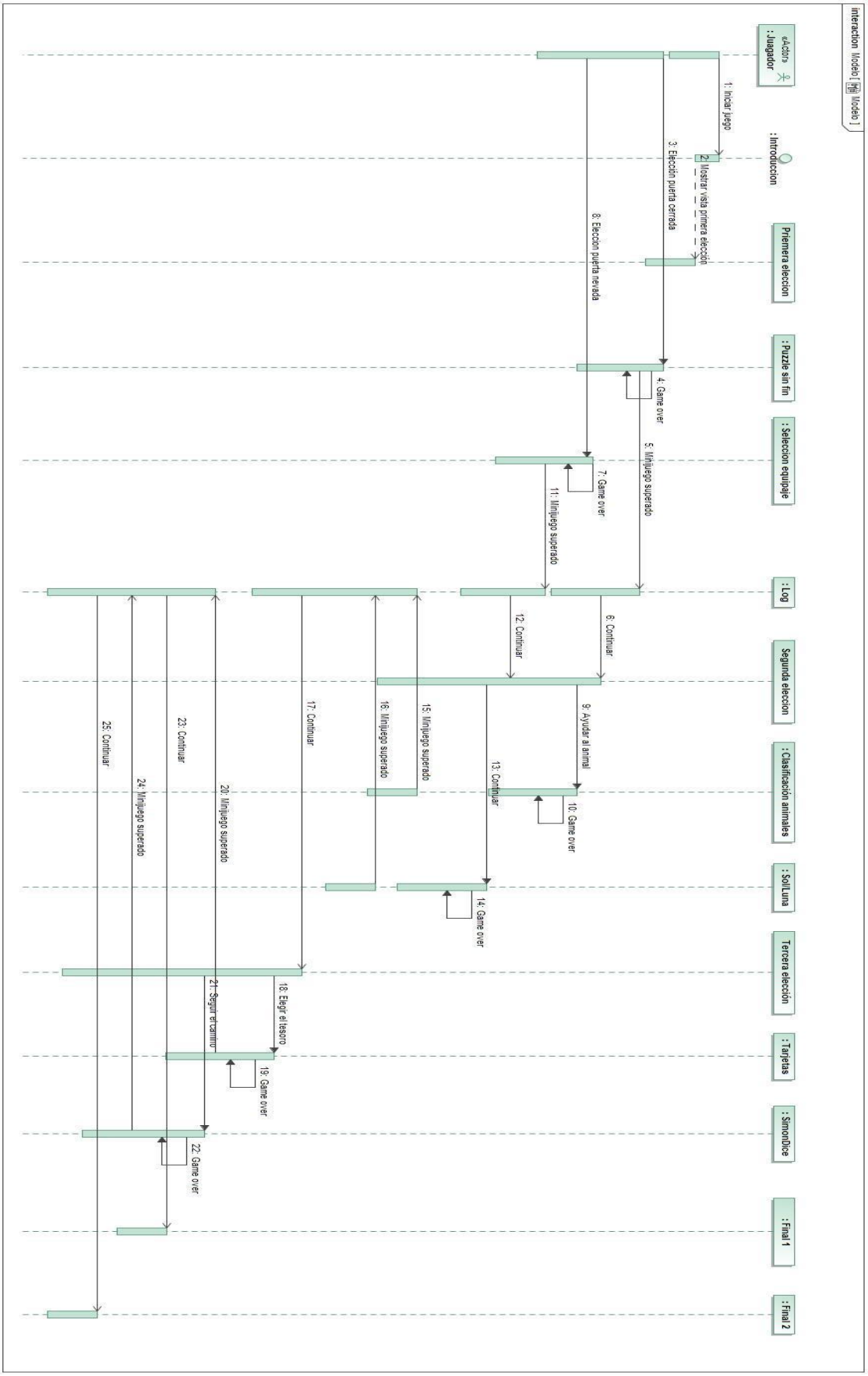
interaction Modelo1 Modelo 1
«Actor»
: Jugador
: Introduccion
Primera eleccion
: Puzzle sin fin
: Seleccion equipaje
: Log
Segunda eleccion
: Clasificación animales
: SolLuna
Tercera elección
: Tarjetas
: SimonDice
: Final 1
: Final 2
1: Iniciar juego
2: Mostrar vista primera elección
3: Elección puerta cerrada
4: Game over
5: Minijuego superado
6: Continuar
7: Game over
8: Eleccion puerta rerada
9: Ayudar al animal
10: Game over
11: Minijuego superado
12: Continuar
13: Continuar
14: Game over
15: Minijuego superado
16: Minijuego superado
17: Continuar
18: Elegir el tesoro
19: Game over
20: Minijuego superado
21: Seguir el camino
22: Game over
23: Continuar
24: Minijuego superado
25: Continuar

Figura 7: Diagrama de secuencia del juego

3.2 Minijuegos detallados

Antes y después de cada juego, se da un pequeño contexto dentro de la historia de este, junto con una explicación para que no haya dificultades a la hora de jugarlo. Ya que la edad de los niños a los que está dirigido estos juegos pueden variar, se ha añadido voz al personaje para que sea accesible a aquellos jugadores que aún tengan dificultad para leer. Esta pantalla cambia dependiendo del minijuego que le siga, en el caso de la Figura 8, el contexto y la explicación del minijuego corresponden al juego *Preparamos el equipaje* que se explicará a continuación.

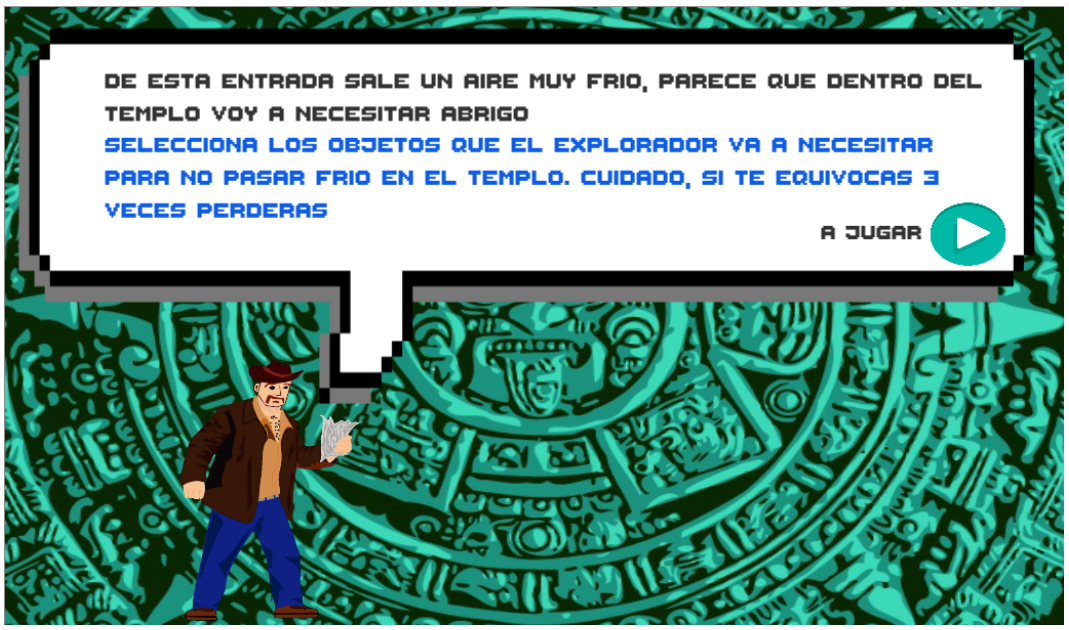

Figura 8: Pantalla consejos

También, al final de cada minijuego aparecerá una pantalla con un pequeño comentario de parte del personaje además de algunos de los indicadores definidos, esta es la pantalla de resultados. Al igual que en la pantalla previa a cada minijuego, la pantalla de resultados mostrada cambiará dependiendo del juego. Por ejemplo, en la mostrada en la Figura 9 los indicadores son el tiempo en realizar la prueba y el número de errores cometidos.

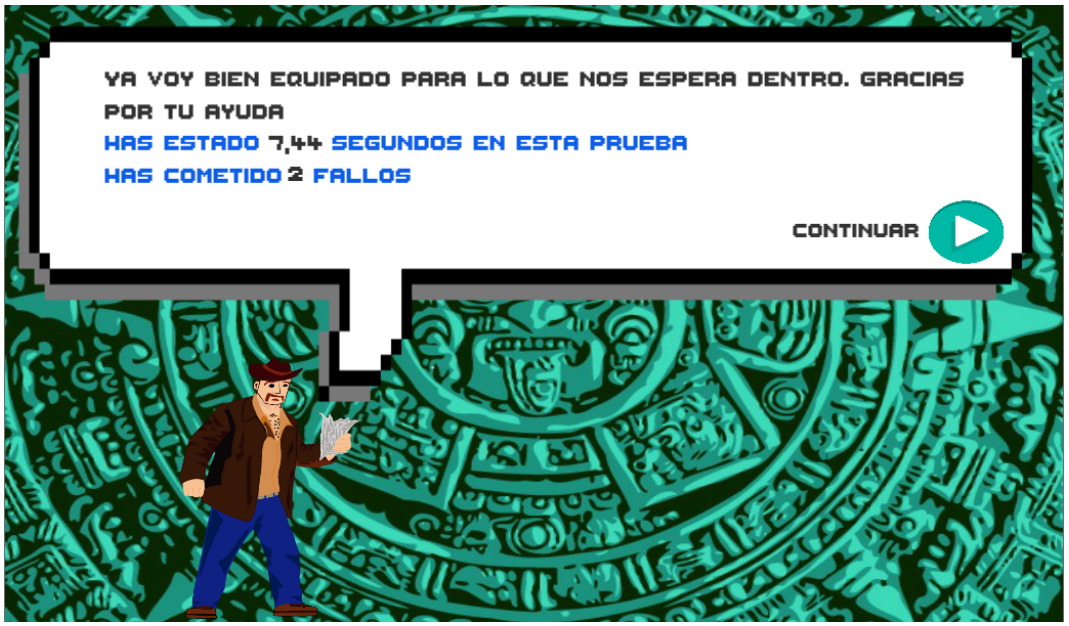

FIGURA 9: RESULTADOS DENTRO DEL JUEGO

3.2.1 Preparamos el equipaje

En este minijuego (Figura 10), el niño deberá elegir qué prendas de ropa son las adecuadas para llevar a un sitio frío, entre un grupo de posibles ítems seleccionables.

FIGURA 10 : CAPTURA DEL JUEGO PREPARAMOS EL EQUIPAJE

Dado que el jugador debe plantearse que vestimenta es la adecuada para combatir al frío, se está tomando en cuenta tanto las funciones ejecutivas de organización y planificación.

Por otro lado, y por la misma funcionalidad del juego, es necesario centrarse en qué elementos son los correctos y evitar cuales son los incorrectos, por eso se hace uso de la función de monitorización

3.2.2 Puzle sin fin

En este minijuego (Figura 11) se presenta un puzzle sin resolver y en el que se dice al jugador que el objetivo es resolverlo. Aunque parece un puzzle sencillo de 9 piezas, cada cierto tiempo, alguna de las piezas colocadas volverá a su posición inicial. Esto se repetirá hasta que el niño pulse sobre el botón terminar donde el juego se dará por concluido.

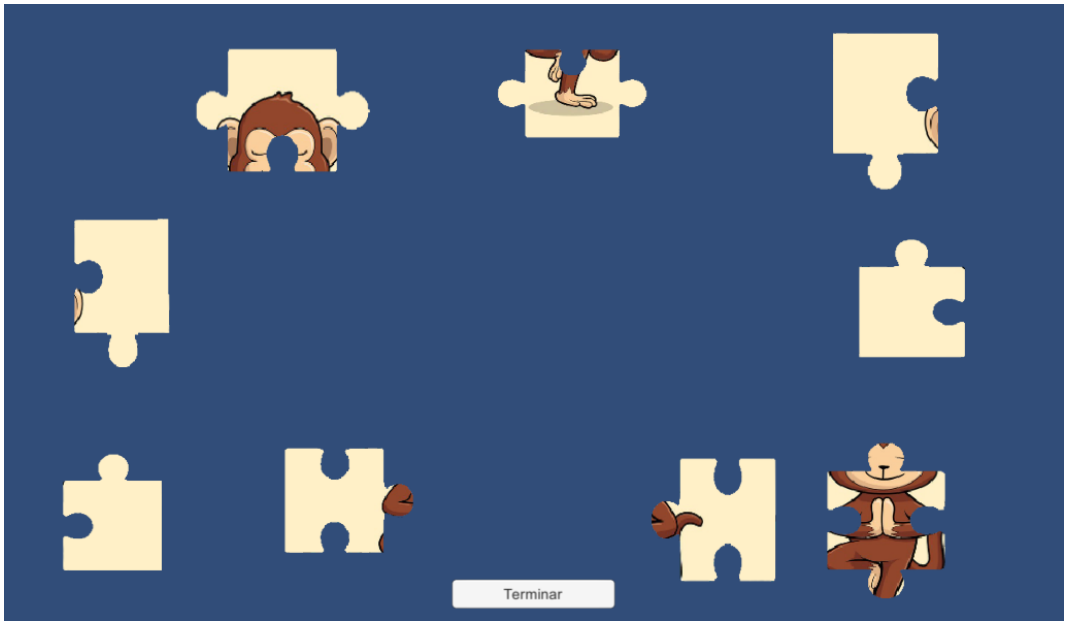

FIGURA 11 : CAPTURA MINIJUEGO PUZLE SIN FIN

Esta tarea ejercita numerosas funciones ejecutivas, ya que, por un lado, activa aquellas relacionadas con la resolución de un puzle, la realización de estos puede favorecer múltiples habilidades y competencias cognitivas. En primer lugar, hace que nos marquemos un objetivo a lograr y, posteriormente, tendremos que poner en marcha mecanismos como el razonamiento y la resolución de problemas, que nos van a permitir conseguir nuestra meta (Callejo, 2020). Las funciones ejecutivas involucradas por lo tanto son las de organización y la planificación

cuando el niño va pensando cómo va a realizar la tarea: comenzar por las esquinas, buscar patrones … Otra de las funciones relacionadas con la resolución de un puzle es la toma de decisiones, ya que una vez planeado, debe comenzar a tomar decisiones simples como por qué pieza empezar, con cual seguir, probar piezas, ver el dibujo etc.

Una vez el jugador se da cuenta de que la tarea no parece acabar nunca, entran en juego otras funciones como el inicio y fin de tareas, ya que para completar el juego deberá darse cuenta de que no existe solución posible, y deberá darse por vencido y pulsar sobre el botón terminar.

La flexibilidad también se toma en cuenta, ya que cuando el jugador comienza la tarea no identifica ningún imprevisto hasta que por primera vez se descoloca el puzle, es aquí donde también entra en juego la monitorización, el niño deberá seguir realizando la tarea y ser capaz de corregir el problema que le ha surgido.

3.2.3 Sol/Luna

En esta mini tarea (Figura 12) se presenta al niño una serie de cartas y deberá responder rápidamente lo que ve. Estas cartas deben representar ítems contrarios, en este caso será el sol y la luna. La dificultad al juego se añade en que las cartas tienen un borde coloreado, si el borde es verde, el jugador deberá decir el ítem representado de manera correcta. Por otro lado, si el borde es rojo deberá decir el ítem contrario. Una vez aceptada una tarjeta, se presentará rápidamente otra distinta. El juego acaba cuando el niño consigue decir correctamente 10 tarjetas seguidas.

FIGURA 12: CAPTURA JUEGO SOL/LUNA

Nuevamente para esta tarea es necesaria una alta concentración, ya que los elementos aparecen tras la pantalla uno tras otro y la respuesta debe ser rápida, por eso se aplica la monitorización.

Debido a que el tiempo es una de los indicadores que se miden en este juego, puede generar un impulso en el niño de intentar responder muy rápido sin tener una respuesta clara, es aquí donde entra también la función ejecutiva de la inhibición.

La respuesta debe ser rápida, pero para darla hace falta tomar una decisión antes, por eso también entra en juego la función de toma de decisiones. Además de esto, el jugador debe mantener la atención en cómo se está realizando la tarea y no dejar de prestar atención, en el caso de cometer un fallo es necesario repetir, de esta forma se enriquece la función ejecutiva de monitorización y anticipación.

3.2.4 Ayuda a los animales

En este minijuego el niño deberá clasificar los animales según si son animales de selva, animales marítimos, aves o insectos (Figura 13). Los animales irán apareciendo en la pantalla y el niño deberá tomar el animal y moverlo al sitio correcto.

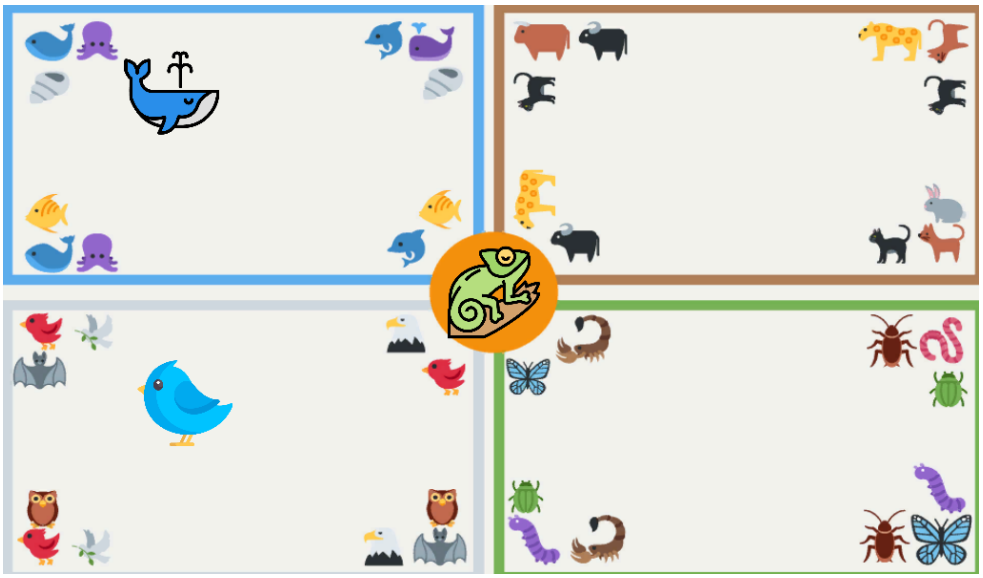

FIGURA 13 : CAPTURA MINIJUEGO AYUDA A LOS ANIMALES

Al igual que con la tarea anterior, se hace un gran trabajo de monitorización al estar concentrado en que animal va a aparecer. Durante el minijuego, el niño deberá observar el animal que ve en pantalla y relacionarlo con una de las cuatro zonas que aparece en pantalla, es por eso porque este juego está muy ligado con la función ejecutiva del razonamiento, ya que para superar el juego será necesario recoger ambas informaciones y ver las conexiones existentes

3.2.5 Memorización de tarjetas

Este minijuego es la adaptación del clásico juego de las parejas (Figura 14). En este juego hay una serie de cartas boca abajo, el jugador levanta una carta y a continuación otra, si ambas tarjetas coinciden se retiran de la mesa, si no vuelven a ponerse boca abajo

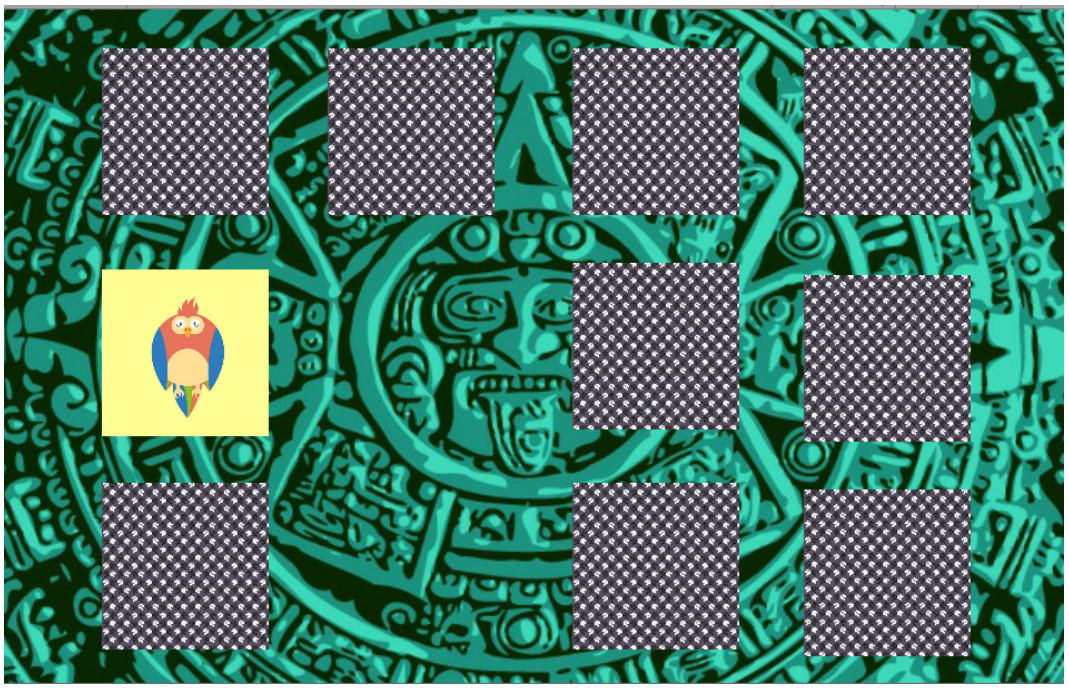

Figura 14 : Captura juego Memorización tarjetas

Con este minijuego se logra trabajar la memoria de trabajo, ya que, para superarlo, el jugador deberá recordar las posiciones de las cartas. Por otro lado, se trabaja la organización, ya que si se quiere superar el juego más rápidamente se pueden seguir algunas estrategias como no levantar la misma carta tras realizar un fallo.

Por último, la toma de decisiones y la anticipación también toman un papel importante dentro del juego, debido que, en caso de duda, el jugador deberá tomar una decisión entre sus posibilidades y deberá anticiparse y crear un plan en caso de fallar.

3.2.6 Simón dice

Este minijuego es una implementación clásica del juego. Consiste en 4 botones de colores que se iluminan para mostrar una secuencia (Figura 15), una vez terminada, el niño deberá replicarla.

Figura 15: Captura juego Simón Dice

Este minijuego utiliza principalmente la función ejecutiva de la inhibición y de la memoria de trabajo. Junto a esta función ejecutiva, también se aplica la función de inhibición, ya que permite regular los actos impulsivos que pueden provocar este juego en los niños, como puede ser pulsar antes de que se acabe la secuencia.

3.3 Arquitectura general

El proyecto se divide en varios componentes, los cuales son:

- El juego completo, es decir, las pantallas de elección junto a los minijuegos.
- Los minijuegos seleccionados por separado, adaptados para su integración alojados en Siette
- Siette, el portal donde se integran los juegos
- Los editores de los minijuegos en el portal

El resumen de los componentes y las relaciones entre sí se puede ver con mayor facilidad en el diagrama de la Figura 16.

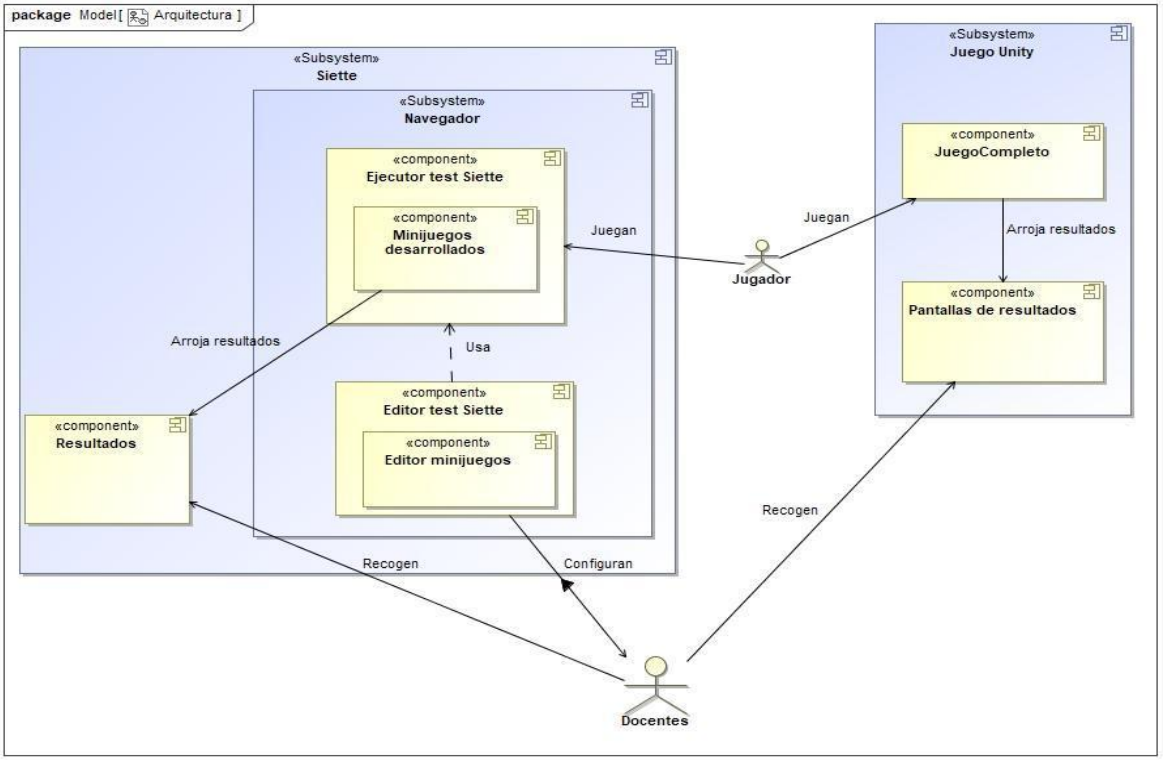

FIGURA 16 : ARQUITECTURA GENERAL

3.3.1 Integración del juego en SIETTE

Para la integración de los minijuegos en Siette hay que realizar varios cambios en la estructura de este. El objetivo es conseguir que el sistema sea parametrizable desde la web, para que de esta forma el profesor pueda ajustar el juego dependiendo de las necesidades del alumno, así como cambiar la temática. Además de esto, se busca mostrar un log que pueda ser analizado por parte del profesorado que haga uso del juego.

Entre estos cambios aparte de añadir la posibilidad de que Unity pueda recibir los parámetros a través de JavaScript ,se encuentra el de la introducción de la llamada a la función *Respuesta Activa.* Esta función es la encargada de comunicar al sistema de SIETTE que el juego ha finalizado y que es el momento de mostrar el log por pantalla.

También se han añadido cambios funcionales en algunos juegos, por ejemplo, en el minijuego ya descrito anteriormente Sol/Luna, se ha eliminado el reconocimiento de voz y se han añadido dos botones que realizan la misma

función. Esto es porque en el juego completo, se utiliza un plugin de Windows que no es compatible al hacer la build del juego para web.

3.3.1 Ejecución de la tarea en SIETTE

Para la ejecución de la tarea en Siette es necesaria una preparación previa. En primer lugar, el test con el minijuego debe estar configurado con los parámetros que el docente haya visto que se adapten al alumno. Después de esto, si se desea, se puede configurar el sistema para que devuelva una corrección a la respuesta Una vez configurado, el niño navega hasta la prueba a partir de un código QR, donde está incluido un identificador del alumno para mantener la sesión monitorizada. Cuando finaliza el juego, la actividad habrá quedado monitorizada y quedará para el estudio de los profesionales.

3.3.2 Cálculo de puntuaciones

Las puntuaciones se obtendrán una vez terminado cada uno de los minijuegos. Para obtener estas puntuaciones se toma en cuenta una serie de indicadores, que irán variando, dependiendo del minijuego. Estos son:

- Tiempo total: es el tiempo que tarda el jugador en completar el juego.
- Fallos: número de fallos cometidos por el jugador durante la prueba.
- Fallos (%): porcentaje de fallos realizados por el jugador respecto al número total de fallos permitidos.
- Tiempo medio de respuesta: tiempo que tarda el jugador entre acciones.
- Tiempo medio entre aciertos: tiempo que tarda el jugador entre un acierto y otro.
- Fallos consecutivos: número de fallos realizados de manera consecutiva por el jugador.
- Aciertos consecutivos: número de aciertos realizados de manera consecutiva por el jugador.

3.3.3 Generación de los resultados

Para la generación de este se han añadido temporizadores y contadores dentro de los scripts que manejan cada minijuego, de tal forma que, al acabar, el minijuego, se realiza la llamada a un método que será el encargado de calcular

los resultados del juego. Dentro del juego en Unity los resultados se muestran en la pantalla de diálogo al acabar el minijuego, como se ve en la Figura 9. Para mostrar los resultados en Siette, es necesario utilizar mecanismos de comunicación entre Unity y el navegador. Este proceso será detallado más adelante.

4

Metodología de diseño

4.1 DESCRIPCIÓN DE LA METODOLOGÍA

Para el desarrollo del videojuego, como para la integración con SIETTE, se ha seguido una metodología de desarrollo ágil, en concreto SCRUM, con ciclos de 3-4 semanas de desarrollo.

SCRUM es un framework o conjunto de técnicas utilizadas para el desarrollo de productos software (Wykowski, T. & Wykowska, J. 2018). Se caracteriza principalmente por la división del trabajo en lo que se conocen como sprints, donde al final de cada uno, se obtiene un producto probable para el cliente.

4.2 CICLO 1. Planteamiento del macro juego y bocetos

Gran parte del primer ciclo de desarrollo fue destinada a la definición de los requisitos del videojuego. Al ser un producto dirigido al público infantil hubo que realizar varios refinamientos de los requisitos, ya que los planteados en un principio podrían no adecuarse a las necesidades de los infantes. Después de varias iteraciones se definió un conjunto de requisitos donde los principales requisitos funcionales recogidos fueron:

- ➢ RF1. El juego debe desarrollar una pequeña historia.
- ➢ RF2. El jugador puede tomar decisiones en la historia.
- ➢ RF3. El jugador puede personalizar al personaje principal
- ➢ RF4. El jugador puede jugar mediante comandos de voz
- ➢ RF5. El jugador puede jugar online
- ➢ RF6. El jugador puede ver un log con su rendimiento en la partida
- ➢ RF7. El jugador puede ver las instrucciones de los minijuegos

También se recogieron algunos requisitos no funcionales:

- ➢ RNF1. Las funciones ejecutivas pueden ser ejercitadas en los minijuegos
- ➢ RNF2. Los minijuegos deben devuelven feedback sonoro a los jugadores

Discutiendo sobre estos requisitos con el equipo de TECHCAT se llegó a la conclusión de que los requisitos RF3, RF4 y RF5, a pesar de que pudiesen ser interesantes para el proyecto, no tenían una prioridad tan alta a diferencia de los demás.

Una vez conseguido un conjunto de requisitos bien definidos, se comenzó con el diseño de los minijuegos. El objetivo era crear un conjunto de minijuegos que pudiesen satisfacer el RNF1 con mayor eficacia. Ni crear muchos porque trabajar con las mismas funciones no iba a aportar nada, ni crear un grupo muy pequeño en el que se intentase agrupar el máximo de funciones en un minijuego.

En un primer momento, se eligió un conjunto de minijuegos mayor al final. Para tener bien definidos los minijuegos, se realizaron casos de uso de cada uno de ellos. En la Figura 17 se puede ver un ejemplo.

Caso de uso

Título	Sol/Luna tarjeta borde verde
Post-condición	El usuario recibe un log del sistema.
Prioridad	Alta
Control de cambios	
Escenario principal	
1. El sistema muestra una tarjeta con el borde verde 1.a El jugador dice en voz alta lo que ve 2. El sistema elige aleatoriamente otra tarjeta 3. El sistema muestra la nueva tarjeta	
1.b El usuario dice en voz alta lo contrario 1.b.1 El sistema muestra una pantalla de Game Over	

FIGURA 17 : CASO DE USO SOL/LUNA CASO POSITIVO

Una vez conseguido un conjunto grande de minijuegos bien definidos el siguiente paso fue comenzar con el diseño del hilo conductor. Antes de esto cabe destacar que durante el final de este primer ciclo ya se descartaron algunos de los minijuegos como el llamado *¿Quién es el mentiroso?*, donde aparecen una serie de ítems y hay que seleccionar los que no cumplen con una característica mostrada, este no fue implementado por ser muy parecido a algunos que ya habían sido aprobados por el equipo de TECHCAT.

Con la elección de la temática y la elaboración del guion el objetivo era crear un pequeño hilo argumentativo que fuese atractivo para los niños, para ello, se investigó sobre cuáles son las temáticas comunes entre los niños y se obtuvo la conclusión de que la mayoría de los niños desarrollan un interés muy grande en algún tema en específico, lo que se conoce como interés intenso, entre los temas más comunes se encuentran los vehículos, la ciencia, la naturaleza o el espacio.

En una primera etapa del diseño, se pensaba tematizar el videojuego en el espacio, donde un astronauta perdido debía superar una serie de pruebas para volver a la nave. En la Figura 18 se puede observar una primera pantalla que iba a servir como puesta en contexto para la historia.

FIGURA 18 : ESCENARIO PRIMERA TEMÁTICA ELEGIDA

Como algunos de los minijuegos no encajaban con la temática, se decidió cambiar a la temática de la naturaleza, donde en un comienzo el personaje iba a realizar un safari por África interactuando con los animales, pero como no se tenía un objetivo claro dentro de la historia se descartó esa idea.

La última idea antes de la definitiva fue la de un turista que descubre por accidente un templo y ve a un ladrón (Figura 19) adentrarse en él, el objetivo era adentrarse en el templo y evitar que el ladrón consiguiera la joya. Finalmente, la idea del ladrón se descartó y se dio lugar a la idea finalmente realizada

FIGURA 19 : BOCETO LADRÓN

4.3 CICLO 2. Implementación del primer bloque de minitareas

Al comienzo de este ciclo, dentro del *Product Backlog* se encontraban tareas de diferente índole. Al existir tanto tareas de diseño a nivel visual como tareas de implementación se decidió dar más prioridad a la implementación.

Antes de comenzar con este segundo ciclo o *sprint*, se realizó algo parecido a un *sprint programming*, se analizó cada una de las tareas que había que realizar para la realización del proyecto y se le otorgó prioridad. En este caso se decidió que las implementaciones más costosas o en las que iban a dedicar mayor investigación tenían más prioridad.

Por esto último, las tareas más costosas, y por lo tanto, más prioritarias se realizaron durante este segundo ciclo, estas fueron el minijuego de *Simón Dice* y el minijuego *Sol/Luna.*

Este último fue un poco más costoso, ya que en primer lugar hubo que investigar sobre cómo se podría implementar el control por voz en Unity, una vez conseguido esto se obtuvo una primera versión, donde después de cada acierto se cambiaba la tarjeta de forma aleatoria. Un error que podría haberse evitado desde los casos de uso es que la propia aleatoriedad provocaba que pudiese salir la misma tarjeta varias veces seguidas, por lo que se decidió que la misma tarjeta no pudiera salir de forma consecutiva. El caso de uso anteriormente mostrado se modificó como se muestra en la Figura 20 para tener en cuenta esta casuística.

Caso de uso

Título	Sol/Luna tarjeta borde verde
Post-condición	El usuario recibe feedback sonoro del sistema.
Prioridad	Alta
Control de cambios	V1.1
Escenario principal	
1. El sistema muestra una tarjeta con el borde verde 1.a El jugador dice en voz alta lo que ve 2. El sistema elige aleatoriamente otra tarjeta entre las restantes 3. El sistema muestra la nueva tarjeta	
1.b El usuario dice en voz alta lo contrario 1.b.1 El sistema muestra una pantalla de GameOver	

FIGURA 20 : CASO DE USO ACTUALIZADO

Antes de comenzar a escribir código, se definió un modelo (Figura 21) que pudiese seguirse en cada uno de los minijuegos implementados. De esta forma, el proceso de programación pudo ser más liviano al ya saberse que estructura había que seguir.

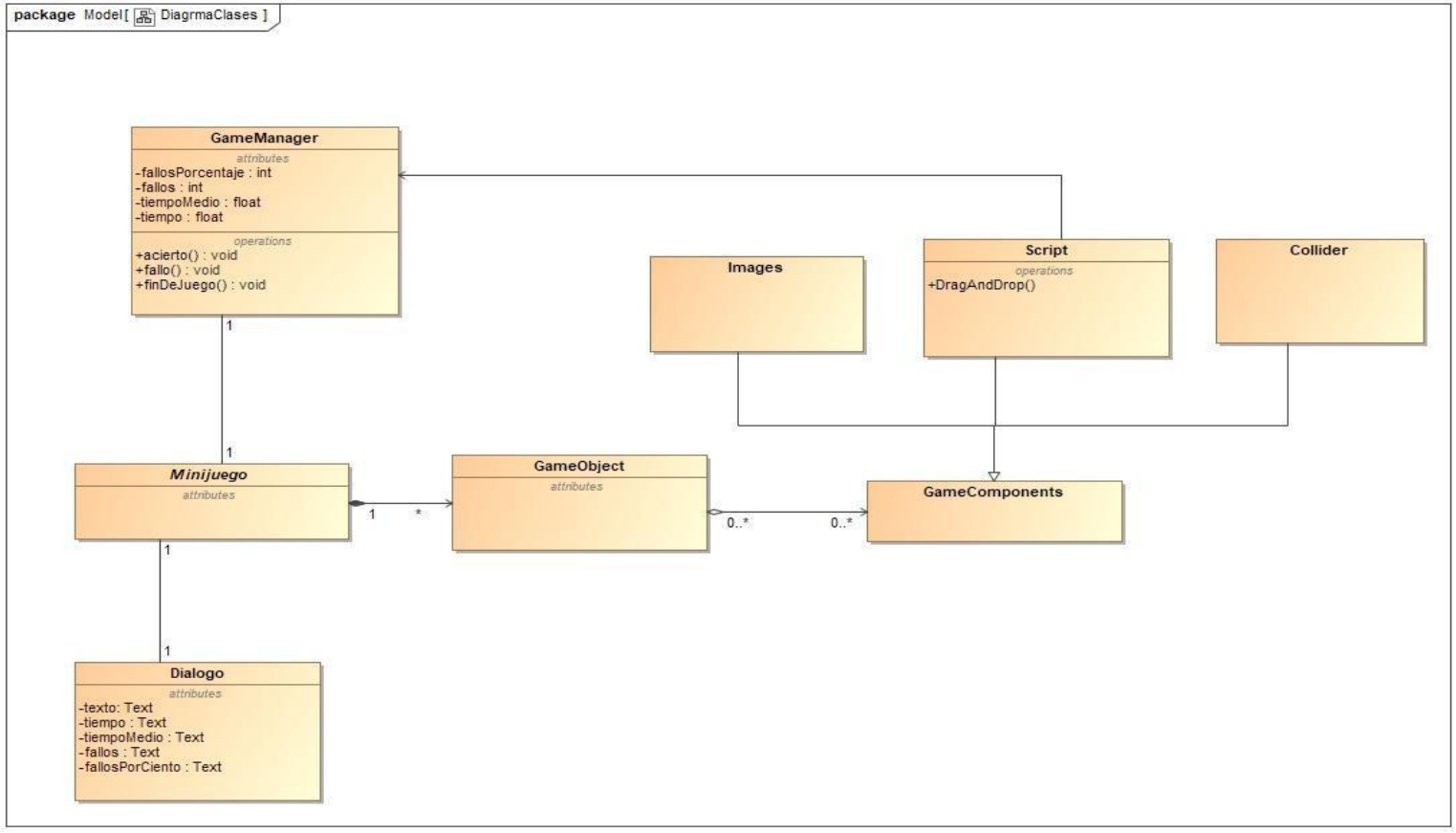

FIGURA 21 : MODELO VIDEOJUEGO

Como puede verse en la figura, cada uno de los minijuegos está compuesto por una serie de GameObjects, una pantalla de Diálogo y un GameManager. Este último es el encargado de ir calculando los indicadores mencionados en apartados anteriores.

A su vez, cada uno de los GameObjects contienen GameComponents, que pueden ser Imágenes, colliders, comportamientos físicos y muchas más opciones que ofrece Unity.

El componente más importante es el Script, que es el encargado de darle a los GameObjects un comportamiento. Es aquí donde la clase GameManager toma sentido, al estar un juego compuesto por muchos GameObjects era necesario una capa intermedia entre estos y lo que ve el jugador tanto mientras está jugando como en el log posterior, ya que es mucho más sencillo recopilar datos de una sola clase, que ir pasando sobre cada uno de los Scripts de cada GameObject y analizar qué ha ocurrido.

El GameManager, es la clase fundamental de cada uno de los minijuegos implementados, ya que es el encargado de la creación y finalización del juego. Una vez el modelo estaba aprobado se comenzó con la implementación.

La implementación del control por voz se dio gracias a un plugin de Windows, una vez instalado, había que configurar una clase llamada KeywordRecognizer. Esta clase es la encargada de recoger e interpretar las entradas por voz. Como se puede ver en el siguiente código, se define un diccionario en el que la clave será un string con el texto que quiera ser reconocido, y una función, de tal forma que cada vez que se escuche el texto se llamará a esa función.

```
1.   private KeywordRecognizer keywordRecognizer;
2.     private Dictionary<string, Action> actions = new
   Dictionary<string, Action>();
3.
4.     void Start()
5.     {
6.         actions.Add("sol", Sol);
7.         actions.Add("luna", Luna);
8.         current = 0;
9.
10.
```

```
11.
12.             rnd = new
    System.Random(System.DateTime.Now.Millisecond);
13.
14.             keywordRecognizer = new
    KeywordRecognizer(actions.Keys.ToArray());
15.             keywordRecognizer.OnPhraseRecognized +=
    VoiceListener;
16.             keywordRecognizer.Start();
17.         }
18.
19.         private void VoiceListener(PhraseRecognizedEventArgs
    speech)
20.         {
21.             actions[speech.text].Invoke();
22.         }
```

El método que recoge la acción de cuando se reconoce la palabra “sol” comprueba si el contenido de la tarjeta se trata del correcto, es decir si la tarjeta es un sol con borde verde o una luna con borde rojo, si es así sonará un sonido de éxito y se obtendrá la siguiente tarjeta, si no se dará feedback y se reiniciará el juego.

```
1.      private void Sol() {
2.          if
    (tarjeta.gameObject.GetComponent<SpriteRenderer>().sprite
    == sol_ok ||
    tarjeta.gameObject.GetComponent<SpriteRenderer>().sprite ==
    luna_no)
3.          {
4.              sonidoExito.Play();
5.              int aux = current;
6.              while (current == aux)
7.              {
8.                  current = rnd.Next(0, 3);
9.              }
10.                cambioTarjeta(current);
11.
    solLunaManager.gameObject.GetComponent<solLunaManager>().ac
    ierto();
12.             }
13.             else {
14.                 sonidoError.Play();
15.
    solLunaManager.gameObject.GetComponent<solLunaManager>().fa
    llo();
16.             }
17.         }
```

Para la implementación del juego de Simón Dice, se utilizó un conjunto de assets obtenidos desde la *Assets Store* que permitiera reproducir sonidos de forma sencilla, ya que en una primera instancia el juego no reproducía ningún tipo de sonido al iluminarse los botones. La reproducción de sonidos durante la ejecución del juego fue algo importante a tener en cuenta, ya que estos sonidos ayudan a memorizar con mayor facilidad las frecuencias a repetir por el jugador.

4.4 CICLO 3. Implementación del segundo bloque de minitareas

Antes de continuar con el desarrollo de las tareas del *Product Backlog* y ya teniendo un modelo bien definido, se decidió diseñar los scripts que iban a ser utilizados en distintos minijuegos. Tras analizar el conjunto de juegos que quedaban por implementar, se decidió que era buena práctica definir un script de selección de elementos y un script que permitiese arrastrar y soltar elementos, que fuese reutilizable en los distintos minijuegos, y así centrar el resto del *sprint* en realizar las tareas específicas que requerían cada minijuego.

Para esto se creó un proyecto muy sencillo de prueba (Figura 22) que consistía en una bola roja en el centro de la pantalla. Sobre esta bola se crearon en una primera instancia los scripts *SelectScript.cs* y *DragAndDrop.cs*.

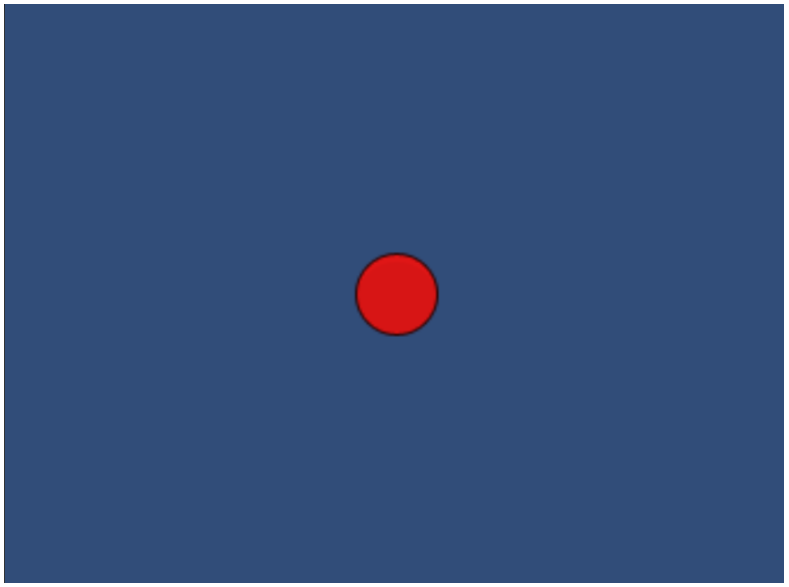

FIGURA 22 : CAPTURA PROYECTO DEMO

Para el script de selección, se utilizó la interfaz de Unity llamada IPointerDownHandler que implementa el método OnPointerDown. Este método recibe como parámetro el evento producido al clicar sobre las instancias de la

clase GameObject que contenga este script. Sobre el proyecto de prueba se implementó una versión sencilla que sirviera como depuración. El siguiente trozo de código hace que al clicar sobre la pelota roja, muestre un mensaje por consola y cambie el color de la pelota a azul.

```
1.     public void OnPointerDown(PointerEventData eventData)
2.     {
3.         gameObject.GetComponent<Image>().color =
   Color.blue;
4.         Debug.Log("click");
```

En el caso de la implementación del script *DragAndDrop* fue necesario el uso de las interfaces aportadas por Unity: *IBeginHandler, IDragHandler* y *IEndDragHandler.* Estas interfaces implementan los métodos necesarios para capturar los eventos de pinchar, arrastrar y soltar. De estos eventos se puede obtener la distancia recorrida por el ratón, que es lo que se usó para cambiar la posición del ítem simultáneamente.

Para el movimiento de la pelota se utilizó el componente del objeto llamado RectTransform. Este componente permite obtener la posición y el tamaño del objeto entre otros muchos valores más.

En una primera instancia en el proyecto de prueba se implementó el código escrito a continuación, que al producirse los eventos movía la bola por la pantalla y además mostraba por consola un mensaje de depuración

```
1.  private RectTransform rectTransform;
2.
3.    private void Awake()
4.    {
5.       transform = GetComponent<RectTransform>();
6.    }
7.
8.    public void OnBeginDrag(PointerEventData eventData)
```

```
9.    {
10.       Debug.Log("comienzo arrastrar");
11.   }
12.
13.   public void OnDrag(PointerEventData eventData)
14.   {
15.       Debug.Log("arrastro");
16.
17.   }
18.
19.   public void OnEndDrag(PointerEventData eventData)
20.   {
21.       Debug.Log("dejo de arrastrar");
22.       rectTransform.anchoredPosition += eventData.delta;
23.   }
```

Al realizar pruebas sobre este código surge un pequeño *bug* que podía llegar a ser bastante molesto y es que el objeto de prueba no seguía de forma precisa la bola, y cuanto más tiempo se estaba arrastrando más disparidad había entre el objeto y el puntero del ratón.

Tras analizarlo se llegó a la conclusión de que ocurre debido a que el objeto de pruebas se encuentra dentro de un objeto *Canvas* de Unity. Estos objetos son los encargados de mostrar por pantalla de forma absoluta lo que se quiera ver, sin importar donde esté la cámara o los elementos dentro del juego. Este objeto canvas está escalado a la pantalla y no puede ser modificado ya que dependerá del sistema, por lo que si el *Canvas* tiene una escala 1.32, el objeto se moverá un 32% más lejos de lo que realmente se está moviendo el ratón

Una vez conocido el problema la solución fue sencilla, cuando cambiamos la posición del objeto será necesario tener en cuenta la escala del *Canvas* donde se encuentra el mismo, quedando el método *OnDrag* arreglado como queda a continuación:

```
1.  public void OnDrag(PointerEventData eventData)
2.  {
3.      Debug.Log("arrastro");
4.      rectTransform.anchoredPosition += eventData.delta / canvas.scaleFactor;
5.
6.  }
```

Una vez teniendo dos de los principales scripts, se decidió que tareas del *Product Backlog* iban a ser las elegidas para formar parte del juego y posteriormente, comenzó su implementación.

Dentro de cada tarea, fue necesario elaborar scripts adicionales específicos de cada minijuego o modificaciones de estos. Estos cambios se detallan a continuación:

➢ Puzle

Este minijuego utiliza una mezcla entre los scripts previamente descritos, ya que cuando el usuario pulsa sobre una pieza y arrastra, esa pieza será almacenada en el *GameManager*.

```
1.  public void OnPointerDown(PointerEventData eventData)
2.  {
3.  gameManager.GetComponent<DesordenarPuzzle>().GuardarPieza(this.gameObje
    ct);
4.  }
```

Este último será el encargado de realizar la funcionalidad de desordenar las piezas. Cada 5 segundos, elegirá una aleatoriamente de entre las piezas almacenadas.

```
1.  void Update()
2.  {
```

```
3.      if (Time.time >= tiempo)
4.      {
5.          int pieza = UnityEngine.Random.Range(0, piezas.Count);
6.          piezas[pieza].GetComponent<Puzzle>().resetear();
7.          piezas.RemoveAt(pieza);
8.          tiempo += 5;
9.      }
10.  }
```

Por último, se añadió el botón terminar, cuya funcionalidad es dar fin al minijuego y mostrar el log por pantalla. Durante esta fase de desarrollo el log aún no se generaba como lo hace en el juego final, simplemente mostraba los datos por consola. En la Figura 23 se puede ver los casos de uso de esta mini tarea.

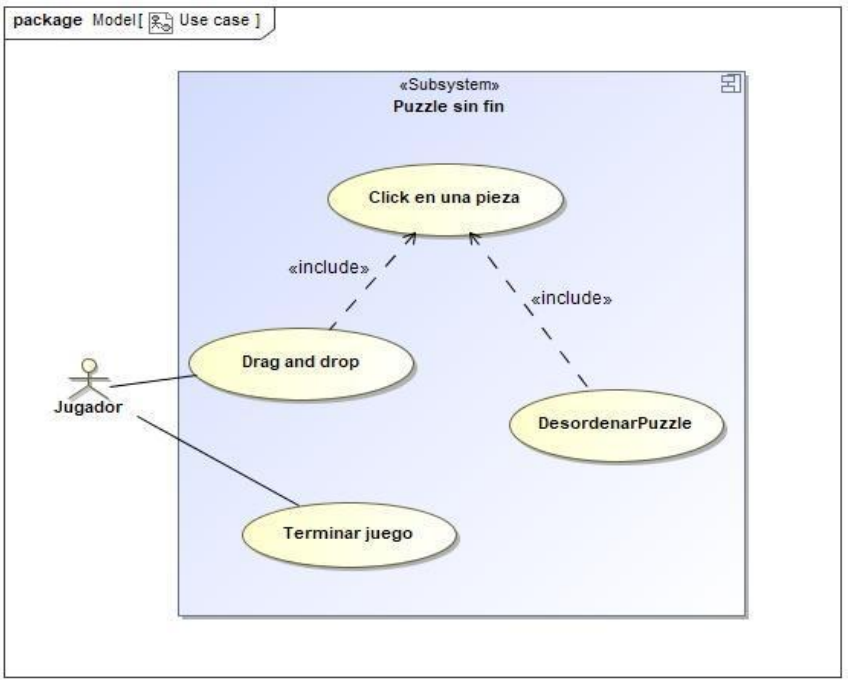

FIGURA 23: CASOS DE USO PUZLE SIN FIN

➢ Preparamos el equipaje

Para la realización de esta tarea no fue necesario añadir mucha más lógica al script de selección. Lo único que había que añadir era que el juego pudiese distinguir qué objetos son los correctos y cuáles no. Para eso se usó el sistema de tags de Unity. Este sistema permite asignar a las distintas instancias de *GameObjects* una etiqueta que luego puede ser usada dentro del juego.

Cuando el jugador pulsa sobre algún ítem éste será el encargado de decir si es un ítem válido o no, y le pasará el resultado al *GameManager* que será el encargado de contabilizar los aciertos o fallos y de poner fin al juego cuando se han completado con éxito o si se ha alcanzado el límite de fallos. En la Figura 24 se puede ver los distintos casos de uso del minijuego.

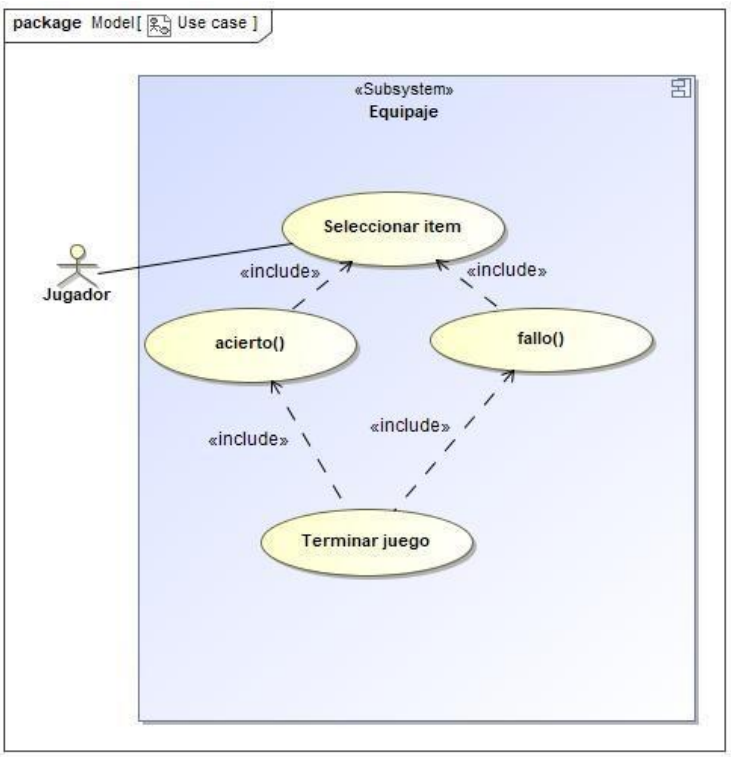

FIGURA 24: CASOS DE USO PREPARAMOS EL EQUIPAJE

En el siguiente fragmento de código se puede ver el uso de los tags dentro de la modificación del script de selección.

```
1.  public void OnPointerDown(PointerEventData eventData)
2.  {
3.      if (gameObject.tag == "Nieve")
4.      {
5.          audioData.Play();
6.          Destroy(gameObject, 1.3f);
7.          viajeManager.gameObject.GetComponent<viajeManager>().acierto();
8.
```

```
        }
        else {
            audioData.Play();
            viajeManager.gameObject.GetComponent<viajeManager>().fallo();
        }
    }
```

➢ Ayuda a los animales

Durante la realización de esta tarea, hubo que realizar algunas modificaciones en el script de *DragAndDrop*, además de generar un script adicional que controlarse donde se soltaban los elementos.

Comenzando con el primero de los scripts, una de las condiciones que se tomaron en cuenta durante la definición del minijuego y sus casos de uso es que una vez el jugador soltase un ítem en una de las casillas, ya no podía volver a recolocarlo. Esta funcionalidad no se encuentra en el script base, así que se realizó una refactorización en busca de la solución.

La primera solución dada fue la más sencilla, añadir un booleano como una flag para indicar al ítem que se puede mover, y cuando este se soltase, desactivarlo.

```
public void OnBeginDrag(PointerEventData eventData)
{
    flag = true;
}

public void OnDrag(PointerEventData eventData)
{
    if (flag) {
        rectTransform.anchoredPosition += eventData.delta;
    }
}

public void OnEndDrag(PointerEventData eventData)
{
```

```
15.     flag = false;
16. }
```

Más adelante durante la finalización del sprint, además de probar cada uno de los minijuegos se investigó y buscó formas más eficientes para mejorar el código. Se encontró que, las interfaces anteriormente mencionadas de *IBeginDragHandler*, *IDragHandler* etc. Utilizaban un raycast de manera interna para acceder a los elementos. En Unity, un raycast no es más que un rayo invisible infinito que es capaz de detectar los objetos con los que colisiona.

En Unity es posible indicar a los objetos que bloqueen los raycast, es decir, hacerlos indetectables a ellos. Por esto se decidió hacer la siguiente refactorización de código ahorrando un método, una implementación y una variable

```
1.  public void OnBeginDrag(PointerEventData eventData)
2.  {
3.      canvasGroup.blocksRaycasts = false;
4.  }
5.
6.  public void OnDrag(PointerEventData eventData)
7.  {
8.      rectTransform.anchoredPosition += eventData.delta;
9.  }
```

Una vez que se comienza a arrastrar el objeto se bloquea la posibilidad de que pueda recibir raycast, de esta forma una vez que se suelte no se podrá volver arrastrarse.

Al igual que con el juego de *Preparemos el equipaje*, el *GameManager* será el encargado de contabilizar los aciertos y errores, así como de poner fin al juego cuando sea necesario. En este minijuego, esta clase también tiene la responsabilidad de generar el juego, ya que los animales que se generan en el centro de la pantalla son elegidos de forma aleatoria.

Al haber cuatro slots donde el niño puede depositar los animales que van apareciendo, deberá haber cuatro recursos diferentes de los que se van tomando los animales. Cada vez que se produce un acierto o fallo se llama a la función *SiguienteAnimal()*, como se ve en el código a continuación, esta función genera primeramente un número aleatorio del 1 al 4 y lo almacena en una variable llamada tg que indica el tipo de animal que se va a generar. Seguido a esto se genera otro valor aleatorio para tomar uno de los animales del tipo correspondiente almacenados en listas dentro del *GameManager.*

```
1.  private void SiguienteAnimal()
2.      {
3.          tg = UnityEngine.Random.Range(0, 4);
4.          int aux;
5.          switch (tg)
6.          {
7.              case 0:
8.                  aux = UnityEngine.Random.Range(0, terrestres.Count);
9.                  siguiente = terrestres[aux];
10.                 terrestres.RemoveAt(aux);
11.                 break;
12.             case 1:
13.                 aux = UnityEngine.Random.Range(0, mar.Count);
14.                 siguiente = mar[aux];
15.                 mar.RemoveAt(aux);
16.                 break;
17.             case 2:
18.                 aux = UnityEngine.Random.Range(0, aves.Count);
19.                 siguiente = aves[aux];
20.                 aves.RemoveAt(aux);
21.                 break;
22.             case 3:
23.                 aux = UnityEngine.Random.Range(0, insectos.Count);
```

```
            siguiente = insectos[aux];
            insectos.RemoveAt(aux);
            break;
        }
        tiempoMedio = Time.time;

    }
```

Como se puede observar, el resultado se almacena en una variable llamada siguiente, esta variable se usa justo después de la llamada al método, cuando se instancia un nuevo elemento con el método *Instantiate*

```
1.  GameObject clone = Instantiate(prefab, position, Quaternion.identity,
    father.transform);
2.          clone.gameObject.GetComponent<Image>().sprite = siguiente;
3.          clone.GetComponent<CanvasGroup>().blocksRaycasts = true;
4.          clone.tag = tags[tg];
```

Este método recibe el elemento que quiere que se instancie, la posición inicial, en mi caso al ser fija está almacenada en una variable, la rotación del ítem, en este caso *Quaternion.identity* corresponde a que no haya rotación, y la posición del padre, en este caso el Canvas que hace que se muestre por pantalla.

Se vuelve a utilizar los tags para poder asignar a cada ítem un identificador que sirva para su clasificación. Estos identificadores serán los mismos que los que tienen los slots en donde se pueden depositar los ítems. De hecho, es el script del slot donde se realiza la comprobación de si se ha acertado o no, y quien se encarga de llamar al *GameManager* como se puede comprobar a continuación.

```
1.  public void OnDrop(PointerEventData eventData)
2.  {
3.      if (eventData.pointerDrag != null) {
4.          if (eventData.pointerDrag.tag == gameObject.tag)
```

```
5.          {
6.              acierto.Play();
7.              eventData.pointerDrag.GetComponent<CanvasGroup>().blocksRaycasts =
    false;
8.
    animalesManager.gameObject.GetComponent<animalesManager>().acierto();
9.          } else
10.         {
11.             fallo.Play();
12.
    animalesManager.gameObject.GetComponent<animalesManager>().fallo();
13.         }
14.     }
15.
16.  }
```

El script implementa el interfaz IDropHandler que implementa el método OnDrop() capaz de capturar cuando un objeto se ha soltado dentro de su collider, comprueba el tag y si ambos coinciden se llamará a la función acierto() del GameManager, si no llamará a la función fallo(). Una vez se hayan generado 15 elementos la evaluación estará lista y se finalizará el minijuego. Todo este esquema puede verse más claramente en la Figura 25 donde se muestran los casos de uso.

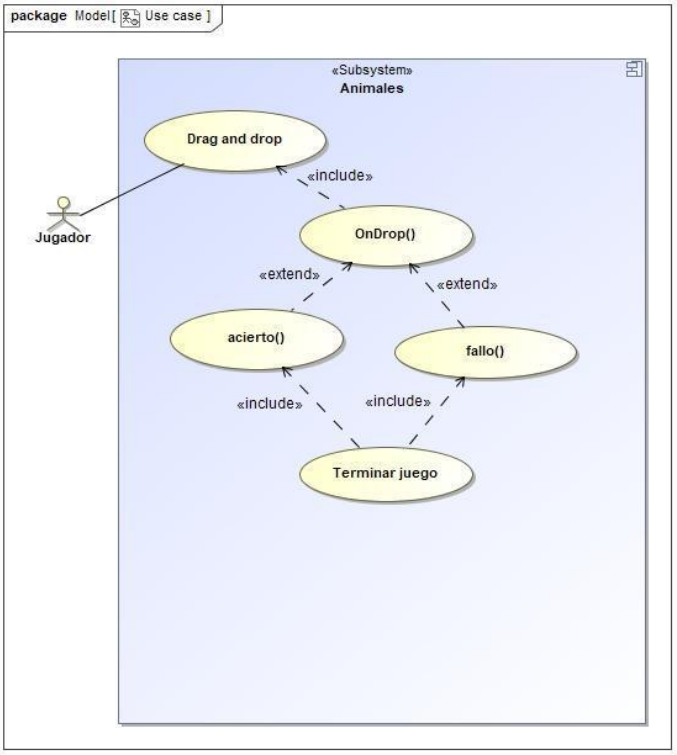

FIGURA 25 : CASOS DE USO AYUDA A LOS ANIMALES

➢ Memoria tarjetas

Para el último de los minijuegos implementados durante este ciclo se usó como base el script *SelectScript.* Cuando una tarjeta es seleccionada se voltea. Para hacer esta implementación cada objeto tarjeta guarda una imagen adicional además de la que muestra. Como se puede ver en el fragmento de código, el método *voltear()* almacena la imagen que se está mostrando en ese momento y muestra la que estaba almacenada:

```
1.  public void OnPointerDown(PointerEventData eventData)
2.  {
3.      voltear();
4.  
    gameManager.gameObject.GetComponent<TarjetasController>().GuardarTarjeta(g
    ameObject);
5.  }
6.  
7.  public void voltear() {
```

```
8.        Sprite imgAux = gameObject.GetComponent<Image>().sprite;
9.        gameObject.GetComponent<Image>().sprite = img;
10.       img = img Aux;
11.
12.   }
```

Como se puede ver, cada vez que se voltea una tarjeta se almacena en el *GameManager* para que, cuando se voltee una segunda tarjeta, se pueda comprobar que ambas tarjetas son iguales

```
1.   public void GuardarTarjeta (GameObject tar)
2.    {
3.       if (tarjeta1 == null) {
4.          tarjeta1 = tar;
5.       } else
6.       {
7.          CompararTarjeta(tar);
8.       }
9.
10.   }
11.
12.   private void CompararTarjeta(GameObject tar)
13.   {
14.      tarjeta2 = tar;
15.      if (tarjeta1.gameObject.GetComponent<Image>().sprite.name ==
      tar.gameObject.GetComponent<Image>().sprite.name) {
16.         esperar = true;
17.         tiempo = Time.time + 1;
18.      } else
19.      {
20.         esperar2 = true;
21.         tiempo = Time.time + 1;
```

```
22.     }
23. }
```

La variable tiempo en este *Manager* no hace referencia al tiempo de juego, es utilizada para que las tarjetas volteadas se vean durante un segundo. Cuando ese segundo ha pasado, si se ha acertado las tarjetas se eliminarán del tablero, habrá un feedback sonoro y volverán a voltearse. Estas funciones están recogidas en los casos de uso del minijuego como se puede ver en la Figura 26. Cuando se llega a emparejar todas las tarjetas, se concluirá el juego

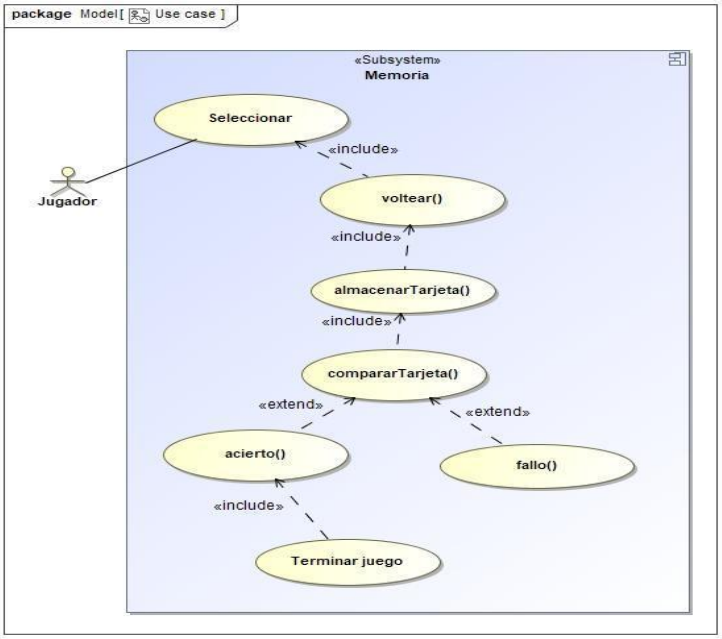

FIGURA 26 : CASOS DE USO JUEGO MEMORIA

Desde el comienzo hasta el final del ciclo, cada una de las mini tareas han pasado por refactorizaciones tanto funcionales como a nivel visual, un ejemplo de esto es el juego de *Ayuda a los animales*, que en un principio iba a mostrar un conjunto de animales fijos desde el comienzo de la partida (Figura 27), pero esto producía que los slots donde debían moverse los animales quedan muy pequeños, además de reducir la complejidad. Junto a este cambio, se pensó que también podía introducirse el indicador de tiempo medio de respuesta.

Figura 27: Primera versión juego Ayuda a los animales

Como se ha comentado anteriormente, al comienzo de esta fase se deliberó que minijuegos del *Product Backlog* iban a ser implementados y cuales se quedaban fuera. A pesar de esto, durante la fase de desarrollo llegaron a desarrollarse algunos minijuegos que finalmente se descartaron. Uno de ellos (Figura 28) consistía en clasificar una serie de ítems por una característica específica, como su color, su forma o su número. El objetivo de esto era tener un juego que se centrara exclusivamente en la función de inhibición, ya que el jugador debía fijarse cuál era la característica por el que debía clasificarlo e ignorar las demás.

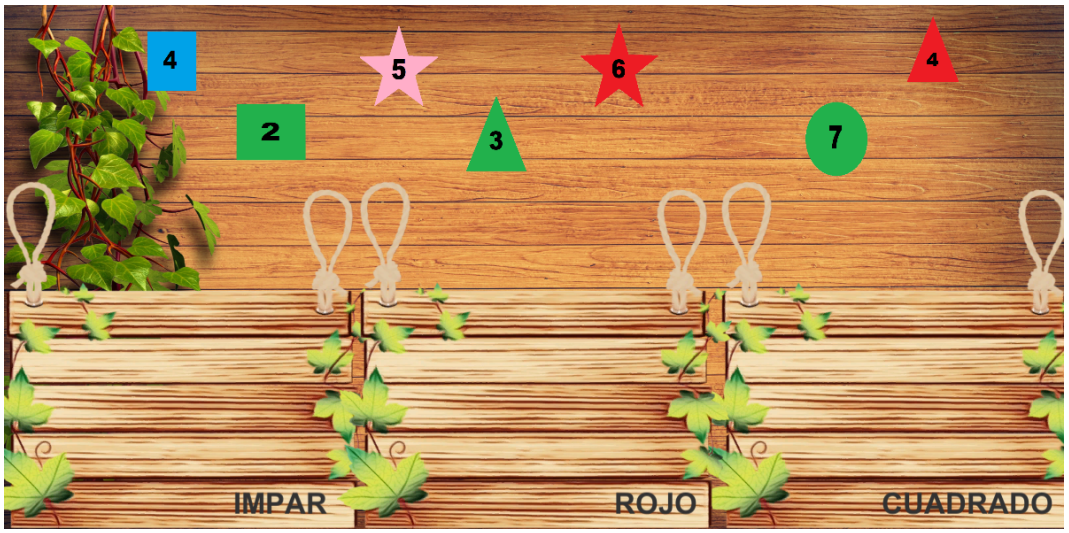

Figura 28: Juego descartado

Finalmente fue descartado en mitad de la implementación debido a que seguía las mismas mecánicas que el minijuego *Ayuda a los animales*

4.5 CICLO 4 Generación de las pantallas de resultados y de decisiones

Este cuarto sprint fue el más corto de toda la fase de desarrollo ya que duró 2 semanas, esto es que se decidió en el *Sprint planning* que había que acabar el juego completo antes de comenzar con la integración de Unity. Para conseguir este objetivo, había que realizar las pantallas de introducción y de decisiones, así como la refactorización de la pantalla de decisiones para darle un resultado más estético.

La creación de las pantallas de decisión, así como el personaje y los escenarios fueron realizados con el programa *Adobe Illustrator*. Al tener cierta experiencia anterior con este programa, la tarea no pasó de la primera mitad de la primera semana del primer sprint.

El resto del sprint se utilizó para refinar qué indicadores se iban a utilizar en cada juego, y para integrar las pantallas realizadas en el juego, incluir diálogos, las pantallas de consejo previamente mostradas etc.

4.6 CICLO 5 CAMBIOS PARA LA INTEGRACIÓN EN SIETTE

Esta fase del desarrollo estuvo pensada para realizar la integración en SIETTE. En primer lugar, había que separar los minijuegos del juego completo en proyectos diferentes para exportarlos de manera individual.

Una vez ya se tenía cada minijuego en proyectos separados, el siguiente paso era instalar WebGL (Figura 29) en cada uno de ellos. WebGL es una opción de construcción de Unity que permite incrustar contenido de Unity en el explorador web, y permite la comunicación entre Javascript incrustado en el HTML y el juego.

FIGURA 29 : ICONO WEBGL

Para realizar la comunicación entre Unity y Javascript se ha seguido el manual que se encuentra en la documentación de Unity. Cuando se hace la construcción del juego, se genera un archivo HTML que es el que alberga el juego. En el script del archivo se genera la instancia del juego, una vez se obtiene la instancia, la función *SendMessage* es la que logra la comunicación con Unity

Esta función recibe 3 parámetros, el primero corresponde al nombre del *GameObject* del juego al que queremos pasarle datos, en nuestro caso será el objeto *GameManager.* El segundo de los parámetros será la función que recibirá los datos y el tercero los datos que les quiere pasar.

Por otro lado, fue necesaria una refactorización en el código para que la evaluación realizada sea recogida por el navegador. En la build del juego para ordenador, el propio *GameManager* era el encargado de recoger los datos sobre los aciertos, fallos, tiempo de juego … que se iban produciendo durante el juego para luego pasarlo a la clase *Diálogo*. Debido a que ahora la muestra del log se produce fuera del juego, es decir, es necesario pasar datos desde Unity a navegador, se ha creado un nuevo *GameObject* llamado Log, que se encarga de recoger y mostrar los datos en el navegador

Para el paso de parámetros se ha definido una función javascript *setParameters()* que es a la que se llama desde el entorno SIETTE. El código HTML del juego construido para cada juego seguiría un esquema similar a este:

```
<script>
      var unityInstance =
UnityLoader.instantiate("unityContainer",
"Build/AnimalesWeb.json", {onProgress: UnityProgress});

        function setParameters(tematicas, elementos) {
            unityInstance.SendMessage("GameManager",
"SelectTematica", tematicas);
            unityInstance.SendMessage("GameManager",
"NumElementos", elementos);
        }

    </script>
```

Para la obtención de los resultados y generación de la corrección se llamará a la función evaluación que a su vez llama a la función *OnApplicationQuit()* que se ha tenido que definir en cada minijuego. Esta función hace uso de un script escrito en lenguaje C, que es el encargado de pasar a la función *RespuestaActiva* la respuesta generada y mostrarla por pantalla, quedando el script del código HTML como se muestra a continuación:

```
    <script>
      var unityInstance =
UnityLoader.instantiate("unityContainer",
"Build/SolLunaWeb.json", {onProgress: UnityProgress});
        setTimeout(function() {
unityInstance.SendMessage("GameManager", "SelecTematica",
6);} , 2000);

        function RespuestaActiva(respuesta) {
            window.parent.RespuestaActiva(respuesta)
        }

        function evaluacion() {
              gameInstance.SendMessage('log',
'OnApplicationQuit');
        }
```

```
12.
13.         function setParameters(tematica) {
14.             unityInstance.SendMessage("GameManager",
    "SelecTematica", fondo);
15.             }
16.       </script>
```

La función *RespuestaActiva* llamará a funciones internas del sistema Siette, que recibirán y almacenarán la respuesta. La respuesta deberá seguir un formato establecido, cada indicador a evaluar separado por comas y entre llaves. El objetivo de esto es que internamente se tomen por separado cada indicador y se almacenen por separado en un array. El objetivo de esto se detalla a continuación.

Cuando internamente se realiza todo esto, la página se recarga, pero esta vez se llama a la función *resolver()*

```
1. function resolver(respuesta, correccion) {
2.           play = false;
3.           for(conta=2500; conta<60000; conta+=2500) {
4.                       console.log("respuesta = "+respuesta)
5.                      setTimeout(function()
    {unityInstance.SendMessage("Log", "Solve", respuesta);  },
6.                        conta );
7.                    }
8.         }
```

Esta función pone un *flag* a falso, indicando que no debe comenzar una nueva partida, además envía a Unity la respuesta anteriormente generada, para poder generar una pantalla de resultados (Figura 30) desde el propio juego:

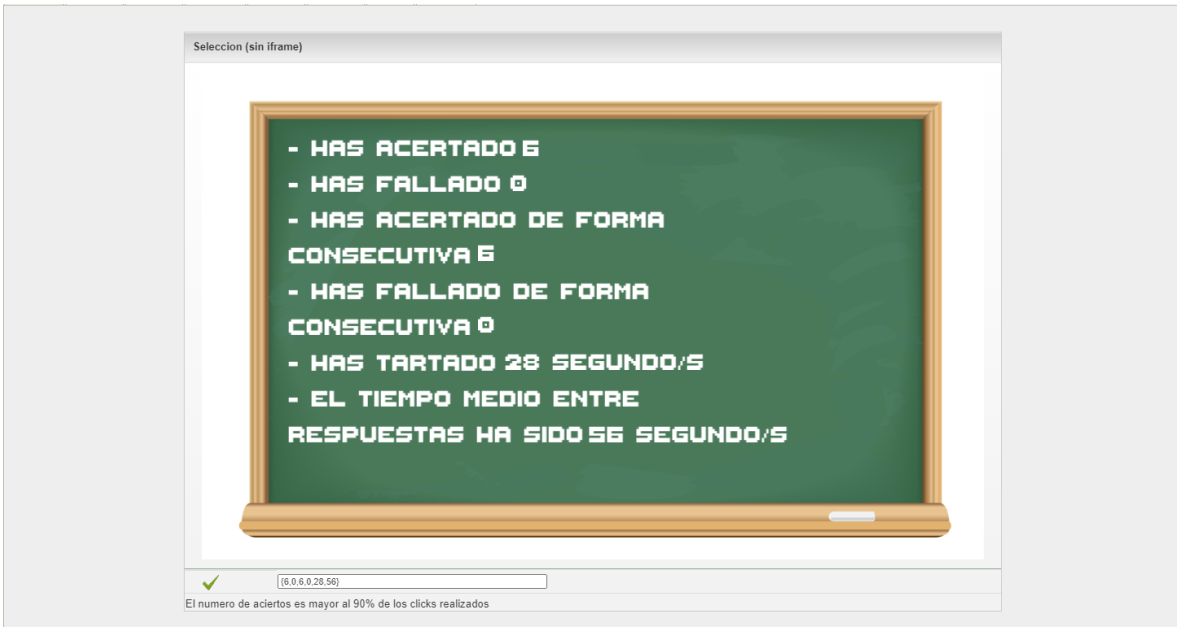

FIGURA 30: PANTALLA DE RESULTADOS EN SIETTE

Además de mostrar los resultados por pantalla, se realiza una corrección desde el propio portal, para ello ha sido necesario definir funciones de evaluación. Las funciones de evaluación son aquellas que, a partir de los resultados generados, van a decidir si una respuesta es correcta o no. Estas funciones han sido adaptadas a cada juego y pensadas para poder medir de forma óptima el rendimiento de cada niño en el juego.

Siette ofrece la posibilidad de definir estas funciones con facilidad, es necesario únicamente conocer en qué orden se han pasado los resultados en la respuesta que se genera al completar el juego

Por ejemplo, en la evaluación definida para el juego *SolLuna (Figura 31)* , una de las soluciones correctas sería que el porcentaje de aciertos sea superior al 90%. Para ello bastará con restar el número de fallos (*answer[1]*) al número de aciertos (*answer[0])* y dividir el resultado por el total de iteraciones que realiza el juego (*answer[7])*

Otra de los indicadores que se deben tener en cuenta en este juego es el tiempo medio entre respuesta, en la tercera función definida se indica que este valor deberá ser menor que 5 segundos

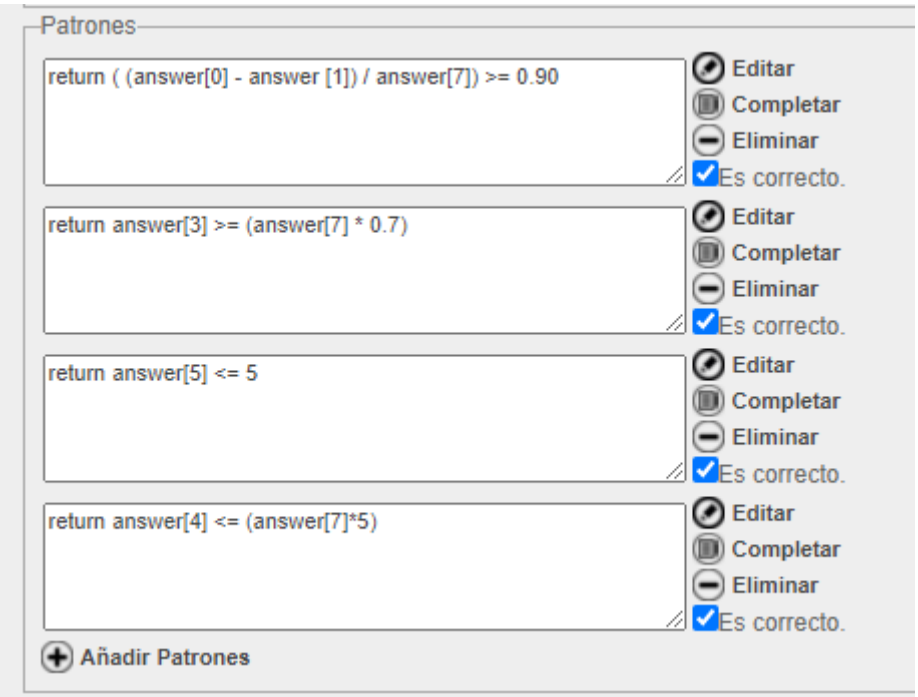

FIGURA 31: FUNCIONES DE EVALUACIÓN SOL/LUNA

En otro de los minijuegos, como en de *Simón Dice*, donde se quiere medir la inhibición, se ha añadido un parámetro a la respuesta, con el número de clics realizados por el jugador. El resultado será positivo si el infante logra un número de clics inferior al número total de iteraciones definidas por el número de aciertos menos el número de fallos (Figura 32). Este resultado indicará si el niño ha sido capaz de controlar los impulsos y si es capaz de hacer un número de clics no muy superior a los estrictamente necesarios

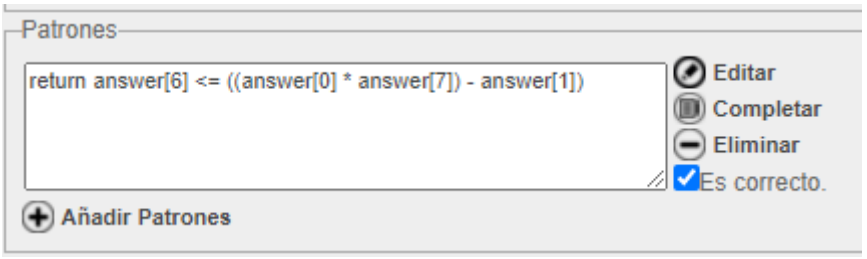

FIGURA 32: FUNCIÓN DE EVALUACIÓN SIMÓN DICE

En el juego, debido a los cambios introducidos, el diagrama de clases anteriormente mostrado cambia ligeramente (Figura 33). La clase *Diálogo* es sustituida por una clase *Log,* liberando al *GameManager* del almacenamiento de los datos y la lógica del cálculo de la evaluación. Adicionalmente, al *GameManager* se le añaden los métodos que son llamados desde javascript y que configuran el juego.

Además de estos cambios generales, se incluyeron cambios funcionales como en el juego *SolLuna* donde se eliminó el reconocimiento de voz por razones de compatibilidad, sustituyéndolo por dos botones. Este cambio no afecta a los objetivos del minijuego.

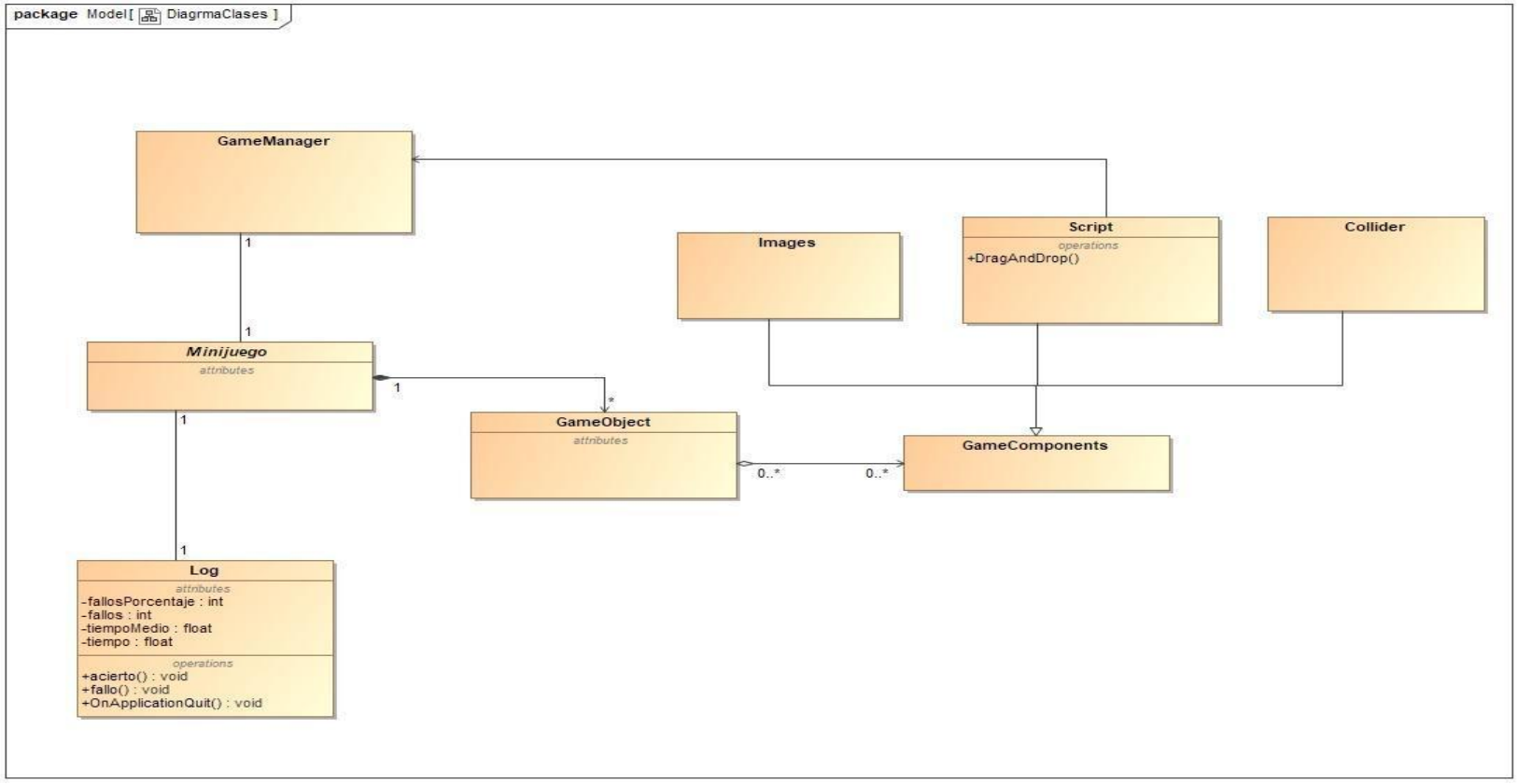

FIGURA 33 : ACTUALIZACIÓN DIAGRAMA DE CLASES

4.7 CICLO 6 Versiones finales y pruebas

Después de la implementación del proyecto y la integración en el portal. Se realizó una pequeña fase de depuración de código y de aseguramiento de la calidad. Para ello se realizaron pruebas *e2e* con el objetivo de que los usuarios tuvieran una experiencia lo más libre de bugs posible

Al final del proyecto se obtienen dos productos:

- Un juego compuesto por una pequeña historia, toma de decisiones y pequeñas tareas.
- Un subconjunto de esas mismas tareas parametrizables y modificables implementadas en el portal SIETTE, para que sea más accesible al profesorado poder seleccionar las actividades que fomenten las funciones ejecutivas que ellos buscan.

5

Conclusiones

La principal conclusión a la que se puede llegar es que los videojuegos son una buena forma de entrenar y enriquecer las funciones ejecutivas ya que permiten realizar actividades y tareas diseñadas por psicólogos y educadores en un entorno lúdico. Para éstos, son un medio no intrusivo de recogida de datos que se pueden analizar para detectar carencias en el desarrollo de los niños, y ofrecer soluciones profesionales para solventarlas.

Por otro lado, se ha observado que, aunque algunas tareas pueden parecer sencillas y triviales, estas pueden suponer un esfuerzo mayor para los niños según sus capacidades, por lo que siempre al trabajar con ellos, es necesario tener muy en cuenta sus necesidades educativas. Esto requiere que los juegos sean configurables en duración, en tamaño de los objetos, en dificultad o en el tipo de destrezas que se requieren para su realización.

Desde el aspecto tecnológico, mencionar la capacidad de la herramienta Unity para la creación de videojuegos. A pesar de ser una herramienta gratuita, ofrece una flexibilidad y versatilidad que la hace competir con

otras herramientas de pago, además de tener una de las comunidades más grandes en internet en lo que respecta a desarrollo

A modo de resumen del trabajo elaborado podemos extraer los siguientes puntos:

- ➢ Se ha estudiado sobre las funciones ejecutivas y sus beneficios, y cómo trasladarlo a los videojuegos. Junto a esto se profundizó un poco más en "el juego imaginativo" como herramienta educativa y su impacto en los niños. Para ello, se ha recogido información sobre distintos tipos de juegos aplicados en la niñez y su comparación con los videojuegos.
- ➢ Se ha desarrollado un guion y unas pantallas para crear ese hilo conductor que lleve la historia
- ➢ Se han desarrollado 6 minijuegos, así como el log y su integración con las elecciones:
    1. *Preparamos el equipaje*: en este minijuego se deberán seleccionar los ítems correctos y evitar los incorrectos, cubre las funciones ejecutivas de la organización, planificación y monitorización
    2. *Puzle sin fin*: se trata de un puzle clásico que se desordena cada cierto tiempo y que trabaja la planificación y organización, además de la toma de decisiones, el inicio y fin de tareas y la flexibilidad.
    3. *Sol/Luna:* van apareciendo figuras opuestas en pantalla, y el jugador deberá decir en algunos casos lo que ve, y en otro el opuesto. Las funciones ejecutivas ejercitadas son las de monitorización, toma de decisiones e inhibición.
    4. *Ayuda a los animales:* consiste en clasificar los ítems con su correspondiente grupo siguiendo alguna característica en concreto. Entra en juego nuevamente la monitorización, además del razonamiento

5. *Memorización de tarjetas:* adaptación del juego de las memorias en el que habrá que conseguir emparejar todas las tarjetas. La principal función ejecutiva que se usa para esta tarea es la de memoria de trabajo.
6. *Simón dice:* el jugador deberá seguir la secuencia de colores indicada. Se trabaja con la inhibición y la memoria de trabajo

- ➢ Para poder evaluar y hacer un seguimiento de la actividad realizada por los niños, para cada minijuego se han generado un conjunto de indicadores sobre el proceso de resolución del juego.
- ➢ Se han integrado 4 de los minijuegos por separado en el portal SIETTE, adaptando cada uno para su correcta implementación. Más en concreto, (i) se han añadido los mecanismos necesarios para la comunicación de Unity y el portal a través de JavaScript. (iii) Se han generado las variables de salida necesarias para construir las funciones de evaluación basada en restricciones con las que se genera la nota que evalúa la actividad en cada juego.

5.1 Futuras mejoras

Algunas de las mejoras que pudieran incluirse serían:

- ➢ Incluir un menú principal en el juego completo, donde el profesorado sea capaz de modificar los minijuegos que vean convenientes para adaptarlos al niño, así como indicar qué indicadores quiere mostrar al finalizar el juego.
- ➢ Añadir más decisiones al juego. Esto supondría añadir también más minijuegos. Otra posible mejora ligada a esto sería que las decisiones dividan los caminos y puedan obtenerse nuevos finales.
- ➢ Creación del personaje. Esta idea permitiría al alumno crear su propio personaje antes de comenzar el juego, haciéndolo personalizable el niño se sentiría más identificado con su personaje con lo que contribuiría al juego imaginativo.

- ➢ Añadir el reconocimiento de voz al minijuego exportado en SIETTE. Esta posible mejora necesitaría bastante tiempo e investigación, pero sería óptimo que la tarea llamada *Sol/Luna* pueda funcionar por comandos de voz.

Además de esto, al ser un producto software existirán muchas posibles mejoras más que se tomarán en cuenta en el futuro

Referencias

Filippetti, V. A. (2011). Funciones ejecutivas en niños escolarizados: efectos de la edad y del estrato socioeconómico. *Avances en psicología latinoamericana*, *29*(1), 98-113 Restrepto, G., Calvachi, L., Cano, I.C. y Ruiz, A.

Rosselli, M., Jurado, M.B., y Matute, E. (2008). Las funciones ejecutivas a través de la vida. *Revista Neuropsicología, Neuropsiquiatría y Neurociencia, 8*(1), 23-46.

Piaget, J. and Gutiérrez Rodríguez, J., 1946. *La formación del símbolo en el niño*. Fondo de Cultura económica.

Yogman M, Garner A, Hutchinson J, HirshPasek K, Golinkoff RM; Committee on psychosocial aspects of child and family health; council on communications and media. The power of play: A pediatric role in enhancing development in young children. Pediatrics 2018; 142

Alonso, N., 2018. *Cuando el niño siente especial atracción por un tema*. [online] Guiainfantil.com. Disponible en: <https://www.guiainfantil.com/educacion/aprendizaje/que-es-el-interes-intenso-en-los-ninos/> [Accedido Mayo 2021].

Callejo, A., 2020. *¿Qué habilidades podemos mejorar si hacemos puzles?*. [online] CuidatePlus. Disponible en: <https://cuidateplus.marca.com/familia/nino/2020/07/23/puzles-mejora-vision-espacial-habilidades-174033.html> [Accedido Mayo 2021].

Castillero, O., n.d. *Las 11 funciones ejecutivas del cerebro humano*. [online] Psicologiaymente.com. Disponible en: >https://psicologiaymente.com/inteligencia/funciones-ejecutivas> [Accedido Mayo 2021].

Red Cenit. 2015. *Entrenamiento de las funciones ejecutivas con videojuegos*. [online] Disponible en: <https://www.redcenit.com/entrenamiento-de-las-funciones-ejecutivas-con-videojuegos/> [Accedido Mayo 2021].

Larrea, B., 2019. *¿Cómo tomamos decisiones? Entre la razón y la emoción - NeuroClass*. [online] NeuroClass. Disponible en: <https://neuro-class.com/tomar-decisiones/> [Accedido May 2021].

Mallén, R., 2016. *Ucrónika: El juego narrativo como estrategia de aprendizaje | El Blog de Educación y TIC*. [online] El Blog de Educación y TIC. Disponible en: <http://blog.tiching.com/ucronika-el-juego-narrativo-como-estrategia-de-aprendizaje/> [Accedido Febrero 2021].

Meneses Montero, M. and Monge Alvarado, M., 2011. El juego en los niños: un enfoque teórico. *Revista Educación*, 25(2), p.113.

Palazuelos, F., 2021. *Qué son los motores gráficos y cuáles son los más populares*. [online] Blogthinkbig.com. Disponible en: <https://blogthinkbig.com/motores-graficos> [Accedido May 2021].

Roldan, M., 2016. *Por qué es tan importante el juego en los niños*. [online] Etapa Infantil. Disponible en: <https://www.etapainfantil.com/importante-juego-ninos> [Accedido Febrero 2021].

Gamelearn: Game-based learning courses for soft skills training. 2017. ▷ *Todo sobre serious games y game-based learning. Ejemplos*. [online] Disponible en: <https://www.game-learn.com/lo-que-necesitas-saber-serious-games-game-based-learning-ejemplos/> [Accedido Mayo 2021].