UNIVERSIDAD DE LAS AMÉRICAS PUEBLA

Escuela de Ingeniería

Departamento de Computación, Electrónica y Mecatrónica

Diseño y desarrollo de prototipo de juego educativo móvil para habilidades de lectoescritura y conteo para niños

Tesis que, para completar los requisitos del Programa de Honores presenta el estudiante

Alexander Díaz Ruiz

ID 160046

Licenciatura en Ingeniería en Sistemas Computacionales

Directora de Tesis: Dra. Zobeida Jezabel Guzmán Zavaleta

Codirectora de Tesis: Dra. Laura Helena Porras Hernández

San Andrés Cholula, Puebla. Primavera, 2022

Hoja de firmas

Tesis que, para completar los requisitos del Programa de Honores presenta el estudiante Alexander Díaz Ruiz, ID 160046

Directora de Tesis

_____

Dra. Zobeida Jezabel Guzmán Zavaleta

Codirectora de Tesis

_____

Dra. Laura Helena Porras Hernández

Presidenta del Comité de Tesis

_____

Dra. Ingrid Kirschning Albers

Secretaria de Tesis

_____

Dra. Laura Helena Porras Hernández

Vocal de Tesis

_____

Dra. Zobeida Jezabel Guzmán Zavaleta

Índice

1. Introducción ……. 7

1.1. Equipo multidisciplinario y aportaciones ……. 7

1.2. Justificación ……. 9

1.3. Objetivos ……. 10

1.4. Alcances y limitaciones …… 10

1.5. Organización del equipo multidisciplinario …… 12

2. Marco de Referencia …… 15

2.1. Ciclo de vida de desarrollo de software …… 15

2.2. Evaluación de usabilidad …… 17

2.3. Patrones de diseño …… 18

2.4. Motor de videojuegos …… 19

2.5. Diseño conceptual de los minijuegos didácticos en esta tesis …… 19

3. Diseño del prototipo ……21

3.1. Integridad de elementos con respecto del juego general …… 21

3.2. El diseño de las secciones desarrolladas en esta tesis …… 24

4. Desarrollo del prototipo …… 39

4.1. Arquitectura del prototipo …… 39

4.2. Clases desarrolladas para los controladores …… 40

5. Evaluación de prototipo …… 57

5.1. Escenas resultantes …… 57

5.2. Evaluaciones …… 58

5.3. Cambios finales …… 62

5.4. Evaluación pedagógica pendiente …… 65

6. Conclusiones …… 67

6.1. Trabajo a futuro ……69

7. Bibliografía …… 71

8. Anexos …… 73

Resumen

Un videojuego con fines didácticos puede ser considerado como una forma de actividad interactiva para favorecer el desarrollo cognitivo en las niñas y los niños. Un juego cognitivo busca desarrollar las capacidades intelectuales del jugador a la vez que fortalece la observación, concentración, y toma de decisiones. Esta tesis define y desarrolla el prototipo de un juego cognitivo para plataformas móviles con sistema operativo *Android*. El objetivo del juego es contribuir con la educación formal de estudiantes entre tercero de preescolar y primer grado de primaria al reforzar habilidades de lectoescritura y conteo. La finalidad es brindar una solución que apoye a la enseñanza de estas áreas cruciales en la educación preescolar y primaria de una manera interactiva, así propiciando su educación de una manera que los incentive a continuar aprendiendo. Para ello, se trabajó de forma interdisciplinaria con estudiantes de las carreras de Pedagogía, Animación Digital y Música, llegando a implementar un prototipo con el motor de videojuegos *Unity*. Se presenta la descripción del proceso de desarrollo de software para tres secciones del ambiente lúdico, así como la descripción detallada del diseño, implementación y evaluación de usabilidad del prototipo de juego propuesto.

Palabras clave: *juegos educativos, educación preescolar, aplicación móvil, Unity*

1. Introducción

EDUPROTEC, A.C. es una asociación civil poblana que tiene como objetivo social favorecer el desarrollo e inclusión social de ciudadanos a través de la aplicación de técnicas de innovación educativa a fin de estimular el aprendizaje de una forma lúdica dirigida a niños, jóvenes y universitarios. Este proyecto inició como una colaboración con dicha organización, el cual tuvo como objetivo el desarrollo de un juego educativo móvil que asistiera a la educación formal de niños entre tercero de preescolar y primer grado de primaria en la enseñanza de lenguaje, matemáticas, y la valoración del patrimonio natural a través de material multimedia interactivo. Debido a la naturaleza multidisciplinaria que conlleva la elaboración de un juego educativo, se extendió una invitación al proyecto a las licenciaturas en Pedagogía, Diseño de Información Visual, Animación Digital, Música, e Ingeniería en Sistemas Computacionales para solicitar su colaboración con el diseño, desarrollo, y evaluación del mismo.

1.1. Equipo multidisciplinario y aportaciones

- Tres estudiantes de Pedagogía bajo la asesoría de la Dra. Laura Helena Porras Hernández, a saber, Marinieves Espinosa Plata, Isabel Maurer Benítez, y Silvia Gabriela Lozada Vega, cooperaron elaborando el marco teórico en el cual se fundamenta el enfoque educativo de la aplicación a desarrollar, así como un storyboard del juego. Este equipo propuso el diseño de un juego de matemáticas que auxiliara al público objetivo en comprender el concepto de número y su relación con el numeral a través del conteo de objetos: un tema que tiene inicio en la etapa de educación preescolar, pero que requiere mayor atención en la educación primaria, de

acuerdo a una entrevista realizada a la Dra. María de los Dolores Lozano Suárez, autora de libros de matemáticas de primaria para la SEP (Espinosa, Maurer, Lozada, 2019).

- Carlos Ángel Grave Rivera y Adrián Holguín López, estudiantes de Diseño de Información Visual y de Animación Digital, respectivamente, ilustraron los personajes, fondos, y todos los recursos visuales originales a ser incluidos en el juego educativo bajo la supervisión del Mtro. Alejandro Ortiz Lima.
- De la licenciatura en Música, el alumno Pedro José Martínez Vieyra ayudó a componer tres piezas originales para la música de fondo y un par de efectos de sonido en la aplicación.
- Mauricio Graciano Álvarez, Itzel Tlelo Coyotecatl, Mauricio Raúl Tenorio Guzmán, y Alexander Díaz Ruiz, estudiantes de la licenciatura en Ingeniería en Sistemas Computacionales, bajo la supervisión de las doctoras Ofelia Delfina Cervantes Villagómez y Zobeida Jezabel Guzmán Zavaleta, aplicaron la ingeniería de software para el diseño de los juegos diseñados teóricamente por el equipo de Pedagogía, generando los diseños de wireframe, los flujos y contenidos de las escenas en la aplicación móvil.
- Trabajo individual del sustentante: Se utilizó el documento de diseño de los juegos de “conteo” y “vocabulario” como guía para la implementación del prototipo de la aplicación móvil. Se integraron los recursos audiovisuales desarrollados por el equipo respectivo. Se llevó a efecto la evaluación con un grupo de interés del prototipo inicial para iterar el proceso de diseño y desarrollo de la aplicación hasta obtener el prototipo presentado en este documento de tesis.

1.2. Justificación

Los alumnos en edad preescolar y de primaria, tienen características particulares de periodos de atención cortos, la necesidad de movimiento y de explorar a través de sus sentidos el mundo que les rodea. Estas características contrastan con la enseñanza pasiva que limita al aprendiente a permanecer sentado a leer, escribir, y escuchar al docente por periodos de tiempo prolongados La ausencia de interactividad en la enseñanza agota el incentivo del alumno por prestar atención a la clase, lo cual propicia su tendencia por distraerse, y a su vez contribuye al desfavorecimiento de su aprendizaje (Febrie, 2018). Por otro lado, el juego es una forma de actividad que favorece el desarrollo cognitivo en las niñas y los niños.

De acuerdo con el Programa de Educación Preescolar, las habilidades mentales se encuentran en un nivel comparable al de otras actividades de aprendizaje durante el desarrollo de juegos complejos involucrando de manera simultánea la atención, imaginación, concentración, estrategias para la solución de problemas, así como el uso del lenguaje (Nuño, Treviño, & Bonilla, 2017). Un videojuego con fines didácticos puede ser considerado como un juego cognitivo, el cual desarrolla las capacidades intelectuales del jugador a la vez que fortalece la observación, concentración, y toma de decisiones manteniendo el interés como la motivación para continuar interactuando a través de éste. Por ende, la ventaja principal que brinda la incorporación del juego educativo al entorno escolar es apreciado por *EDUPROTEC, A.C.*, docentes, alumnos y padres de familia por la posibilidad de hacer realidad el proceso de enseñanza-aprendizaje mediante una herramienta que apoye a todos y a cada uno de los participantes en dicho proceso. Para

ello es relevante considerar en la planeación de situaciones de aprendizaje para alumnos en edad preescolar y edad primaria, contemplar no sólo la forma en cómo aprenden los niños sino también recursos y materiales didácticos pertinentes y relevantes para sus intereses.

1.3. Objetivos

Diseñar y desarrollar el prototipo de un juego educativo para plataforma móvil *Android* basado en el diseño teórico del equipo multidisciplinario constituido por alumnos de las licenciaturas en Pedagogía, Diseño de Información Visual, Animación Digital, Música, e Ingeniería en Sistemas Computacionales. El juego a desarrollar se enfoca para estudiantes entre tercero de preescolar y primer grado de primaria para reforzar habilidades de lectoescritura, matemáticas, y apreciación por el patrimonio natural, y desarrollar un prototipo evaluable basado en el motor de videojuegos *Unity*.

1.4. Alcances y limitaciones

El objetivo final del proyecto multidisciplinario es el desarrollo de una aplicación móvil, la cual es conformada por un menú principal donde el usuario puede registrarse e iniciar sesión, seguido de una introducción que le presenta el mundo en el que toma lugar con el fin de ambientarlo y generar incentivo por usar la aplicación, y un menú de juegos que le ofrece acceso a una selección de juegos didácticos que estimulen su aprendizaje, como se muestra en la Figura 1.

El alcance de esta tesis es diseñar y desarrollar un prototipo de la aplicación para tres secciones: 1) introducción, 2) un minijuego de matemáticas especializado en el conteo

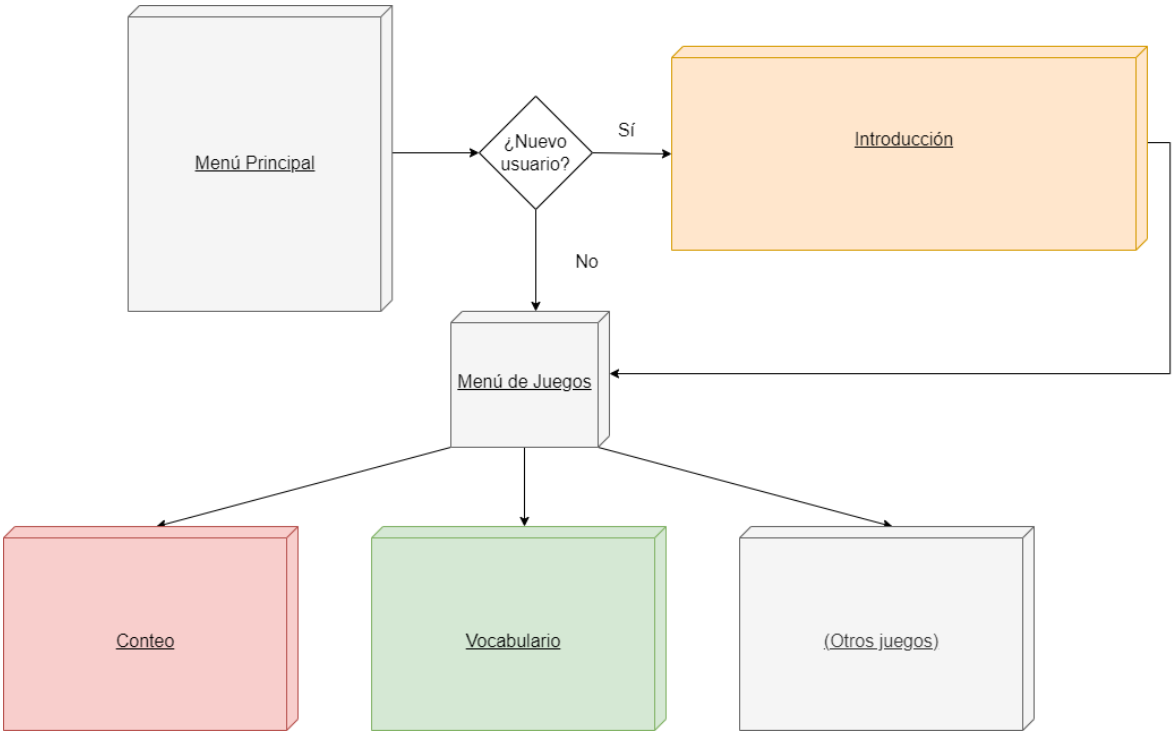


Figura 1. Diagrama de flujo de las secciones que conforman la aplicación móvil al final del proyecto, de las cuales únicamente se incluyen el diseño y desarrollo prototipado de la introducción y los juegos de conteo y vocabulario en el alcance de esta tesis.

(de ahora en adelante denominado “Conteo”), y 3) un minijuego de lectoescritura con énfasis en la construcción de palabras y la relación con su significado (de ahora en adelante denominado “Vocabulario”). Asimismo, este documento muestra las evaluaciones de usabilidad y pedagogía realizadas por la Dra. Guzmán y la Dra. Porras, respectivamente, así como los resultados de un *focus group* en la cual participaron ocho alumnos de la materia *Interacción Humano-Computadora* (LIS3032) impartida por la Dra. Ingrid Kirschning Albers en el periodo Otoño 2020 (en nombre, David Fernando Rosas Chávez, Victor Armando Canales Lima, Fernando López López, Carlos Rubén

Acosta Herrera, Garret Ravelli de la Cruz Tadeo, Eloin Alfredo Puga Flores, Ian Manuel de Gyves Flores, y María Fernanda Flores Luna), cuya retroalimentación conjunta, sugerencias, y mejoras propuestas son revisadas e implementadas para obtener el estado final del prototipo de la aplicación.

Queda fuera del alcance de esta tesis el diseño y desarrollo del menú principal, el menú de juegos, y los juegos didácticos de la aplicación fuera de Conteo y Vocabulario. Asimismo, no está incluida la evaluación pedagógica de la aplicación completa, ni la funcionalidad para recuperar y analizar la información del desempeño de cada jugador, almacenarla en una base de datos en la nube, y generar estadísticas que puedan ser consultadas por los profesores y padres de familia del usuario para identificar sus áreas de oportunidad, y, por ende, brindarle una atención más personalizada a su formación académica sin descuidar de su interés por la aplicación o por su aprendizaje.

1.5. Organización del equipo multidisciplinario

Para satisfacer el objetivo de la tesis mediante el desarrollo de un prototipo de la aplicación móvil, el equipo de Sistemas realizó una inspección del diseño del juego didáctico propuesto por el equipo de Pedagogía. Tras consultar las dudas surgidas, sus expectativas del producto final, y llegar a un acuerdo de los aspectos principales del proyecto, el equipo de desarrollo elaboró una wireframe que especifica las escenas que conforman las secciones de la Figura 1, su contenido, y cómo se relacionaban. Seguido, se emplearon los gráficos provistos por el equipo de diseño para los personajes y los fondos de la aplicación, así como recursos adicionales provistos por bancos de imágenes con licencia de uso libre. La música de fondo fue provista Pedro Martínez, quien forma

parte del equipo multidisciplinario, y las voces de los personajes fueron grabados por miembros del equipo de desarrollo y la Dra. Porras. Como modelo de ingeniería de Software se empleó el Ciclo de Vida de Desarrollo de Software, el cual se explica en el siguiente apartado, así como el patrón de diseño *Modelo-Vista-Controlador* para organizar la arquitectura del proyecto. Para su implementación se utilizó el motor de videojuegos *Unity* versión 2018.4.8f1 (LTS), cuyo prototipo fue sujeto a una evaluación de usabilidad técnica y pedagógica como pruebas de usabilidad, y con la retroalimentación brindada se iteró el diseño para desarrollar un prototipo mejorado.

2. Marco de Referencia

Para iniciar el desarrollo de un proyecto de software, se requiere una planeación y un diseño de la solución propuesta con el fin de asegurar que ésta cumplirá con las expectativas y necesidades del usuario final de una manera fácil, cómoda, e intuitiva. Asimismo, al considerar de antemano los datos que la solución requerirá procesar, cómo se almacenan, y los flujos lógicos que definen su funcionamiento, se agiliza el proceso de desarrollo debido a la minimización de contratiempos que surgen por resolver errores imprevistos, y se garantiza que la solución que se le entregará al usuario final será robusta contra usos inválidos o inesperados. Es por ello que existen procedimientos, marcos, y estándares internacionales para guiar a los desarrolladores de un proyecto a producir software de alta calidad, algunos de los cuales se especifican a continuación.

2.1. Ciclo de vida de desarrollo de software

El ciclo de vida del desarrollo de software (CVDS) es un marco de referencia de desarrollo de software que divide su labor en etapas asimilables que van desde la recolección y análisis de requisitos hasta el despliegue de la aplicación y su continuo mantenimiento (Kneuper, 2017) (véase Figura 2). A continuación, se explica en qué consiste cada una de las etapas del ciclo:

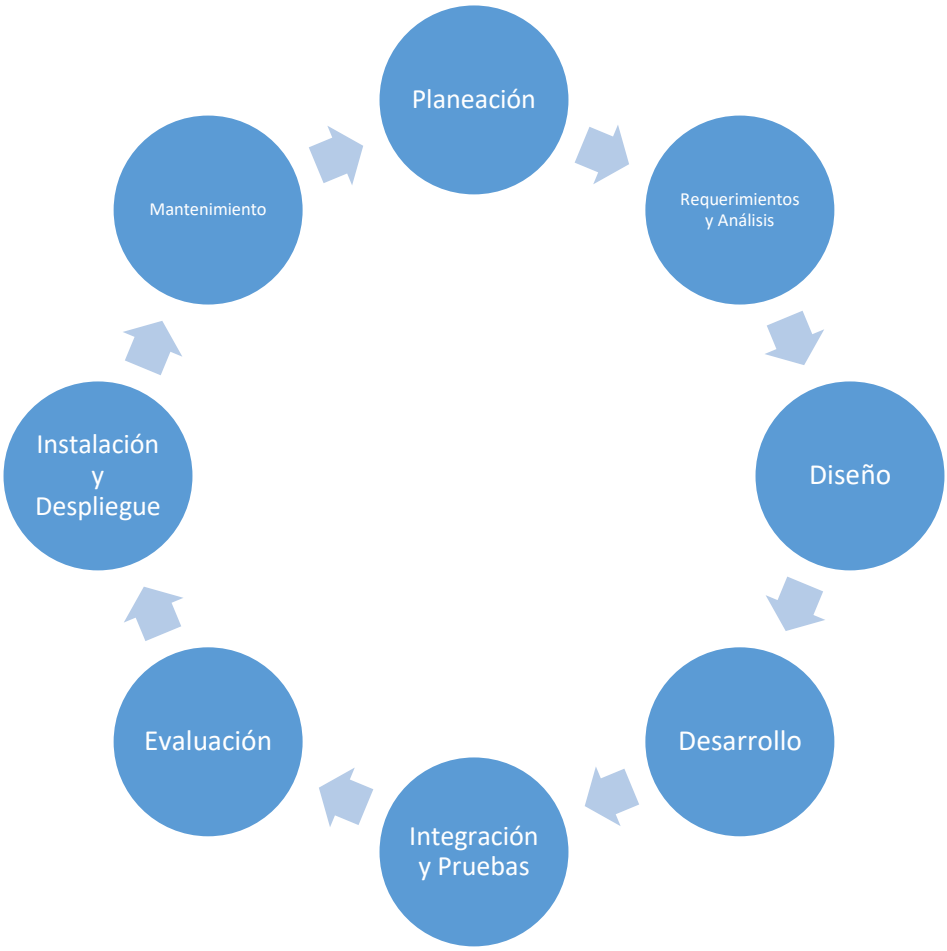


Figura 2. Diagrama de CVDS de 8 fases.

El ciclo inicia con la Planeación, el cual es el proceso por el cual se lleva a cabo un análisis para identificar las acciones a llevar a cabo durante el proyecto a fin de alcanzar los objetivos y metas propuestos. Durante la fase de Requerimientos y Análisis, se reúne información relevante del cliente para desarrollar el producto y asegurar que se alcancen sus expectativas, lo cual involucra la descripción del software a desarrollar, su propósito, y a quién va dirigido. Estos requisitos son entregados al equipo de desarrollo, quien revisa el documento, pregunta dudas, hasta que se tiene una comprensión clara del problema.

La etapa de Diseño involucra el diseño funcional, definiendo la organización del sistema, los elementos estructurales en éste, y su interrelación, así como la experiencia del usuario y la usabilidad de la solución. Durante la etapa de Desarrollo del proyecto se toman las especificaciones del diseño y se implementan en módulos mediante código fuente. En la fase de Integración y Pruebas, los módulos producidos durante la etapa anterior se prueban exhaustivamente para detectar posibles errores lógicos y corregirlos para asegurar la producción de software robusto. Seguido, se integran las unidades, y se verifica que funcionen de manera correcta en conjunto. Durante la fase de Evaluación, los clientes, usuarios finales, y público objetivo prueban la solución para asegurarse de que cumple con sus expectativas y estándares, y que su uso sea cómodo e intuitivo. A continuación, se requiere auxiliar a los usuarios a familiarizarse con el uso del programa durante la fase de Instalación y Despliegue. Por último, la etapa de Mantenimiento aborda errores imprevistos y peticiones para agregar funcionalidades adicionales que pueden surgir, lo cual comienza un nuevo ciclo de desarrollo de software.

2.2. Evaluación de usabilidad

Otro referente importante para el desarrollo de software educativo es la evaluación de usabilidad, la cual mide la calidad de la experiencia del usuario al utilizar el software; es decir, qué tan fácil, cómodo, eficiente, e intuitivo es (Cockton, Department of Health and Human Services, 2013). De no realizar evaluaciones con los interesados de un proyecto de manera periódica, los desarrolladores del mismo se arriesgan a entregar un proyecto que no será bien recibido, y, por ende, encontrarse en necesidad de invertir mayor tiempo y esfuerzo de lo que se tenía previsto para realizar ajustes que se pudieron haber evitado

desde un inicio. Existe diferentes maneras en las que se pueden conducir evaluaciones de usabilidad—supervisadas o no, en remoto o en persona, abiertas a opiniones e impresiones, enfocadas en satisfacción de usuario, o meramente comparativas entre dos propuestas—, cuyas combinaciones producen resultados distintos (hotjar, 2022).

2.3. Patrones de diseño

Existen ciertas estrategias que se utilizan para documentar las soluciones reutilizables de software. Una de estas estrategias es el patrón *Modelo-Vista-Controlador* (MVC) (véase Figura 3). En él se divide una aplicación en una capa de presentación, en el cual se presenta la interfaz de usuario mediante la cual éste interactúa (i.e., *Vista*), una capa de datos que define la estructura en la que se almacenan, así como los métodos relacionados para crear, recuperar, editar, y eliminar los mismos (i.e., *Modelo*), y una capa lógica intermedia (i.e., *Controlador*), cuya función se desempeña en recuperar datos de la *Vista* y del *Modelo*, procesarlos, y actualizar las capas correspondientemente.

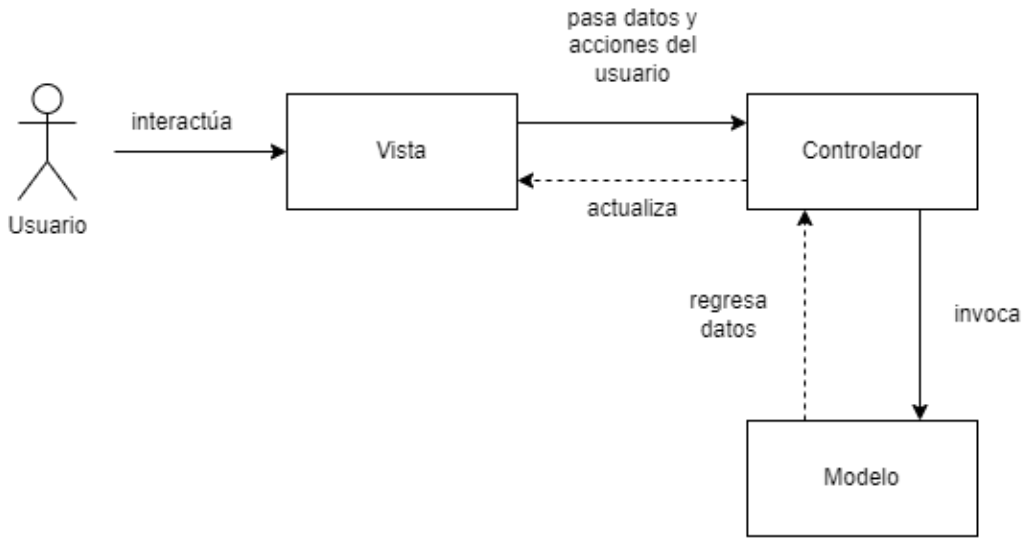


Figura 3. Diagrama del patrón de diseño MVC, y cómo se interrelacionan sus tres capas.

2.4. Motor de videojuegos

Un motor de videojuegos es un programa especializado para el desarrollo de videojuegos, el cual provee fragmentos de código, recursos audiovisuales, y componentes reutilizables desde un inicio, lo cual permite agilizar el desarrollo y subsecuente integración de un videojuego. En este proyecto de tesis se utilizó *Unity* como el motor de videojuegos de elección debido a su popularidad y estandarización en la industria del videojuego, facilidad para exportar a múltiples plataformas desde una base de código, natividad para desarrollar juegos en 2D o 3D, e integración con proveedores de bases de datos en la nube (Unity Technologies, 2022). Sus diferentes vistas o niveles se representan mediante *escenas*, cuyo comportamiento por defecto es extensible mediante una interfaz de programación de aplicaciones (API), la cual permite escribir código en el lenguaje de programación C# para incluir funcionalidad acorde a las necesidades del juego.

2.5. Diseño conceptual de los minijuegos didácticos en esta tesis

2.5.1. Conteo

Tres alumnas de la Dra. Porras, Marinieves Espinosa Plata, Isabel Maurer Benítez, y Silvia Gabriela Lozada Vega, ayudaron a proveer el fundamento pedagógico de la aplicación móvil mediante su propuesta del diseño de un juego didáctico de matemáticas con enfoque en la construcción del concepto de número y el conteo, titulado *Los Números del Bosque*. Para apelar a la apreciación del patrimonio natural, el juego tiene lugar en un bosque de la Sierra Norte de Puebla, en donde habita una pareja de adultos mayores conocidos como el tío Feru y la tía Mati, así como un conejo teporingo llamado Tepo, un colibrí llamado Chip, y demás especies endémicas

de la zona. El juego inicia con los tíos Feru y Mati solicitándole a Tepo que recolecte algunas frutas para que puedan cocinar una tarta, quien a su vez le pide ayuda al jugador para verificar que las frutas cosechadas y sus cantidades sean las correctas (véase Anexo 1).

2.5.2. Vocabulario

Propuesto por el equipo de desarrollo con supervisión del equipo de Pedagogía para asegurar su naturaleza didáctica, el minijuego de Vocabulario busca que el usuario relacione palabras apropiadas de su nivel educativo a su definición mediante un juego de estilo *drag and drop*. El juego le proporciona al usuario la definición de una palabra, cuyas letras se encuentran desordenadas. El usuario tiene la misión de reordenarlas en el espacio provisto, tal que deletreen la palabra cuya definición fue proporcionada. Al terminar de reordenar una palabra, inicia automáticamente otra ronda, tal que se le proporciona una nueva definición, y las letras correspondientes a su palabra. Cada tres palabras denotan un "nivel" en el minijuego, y con cada nivel subsecuente la dificultad de sus palabras aumenta.

Después de haber recibido el documento en donde se establece el fundamento didáctico de la aplicación móvil, consultar dudas, anotar observaciones, y confirmar las expectativas del producto final con el equipo de Pedagogía, se aplicó el diseño de software para representar las escenas que conforman las secciones de Introducción, Conteo, y Vocabulario, así como sus relaciones, contenido, y propósito, mediante wireframes.

3. Diseño del prototipo

3.1. Integridad de elementos con respecto del juego general

Los wireframes de las tres secciones diseñadas (i.e., Introducción, Conteo, y Vocabulario) poseen elementos en común, cuyas representaciones y significados serán explicados en el siguiente subapartado.

3.1.1. Escenas

Cada sección de la Figura 1 representa un grupo de una o más escenas interrelacionadas. Una escena es representada mediante un rectángulo blanco, en cuyo interior se definen su interfaz gráfica de usuario y los elementos que ésta contiene (i.e., etiquetas de texto, imágenes, botones, animaciones, etc.). En la parte superior de cada escena se encuentra una leyenda con su nombre, y en la parte inferior se especifican elementos adicionales y no visuales (i.e., grabaciones de voz, música, efectos de sonido, imágenes de fondo, notas para la implementación) (véase Figura 4).

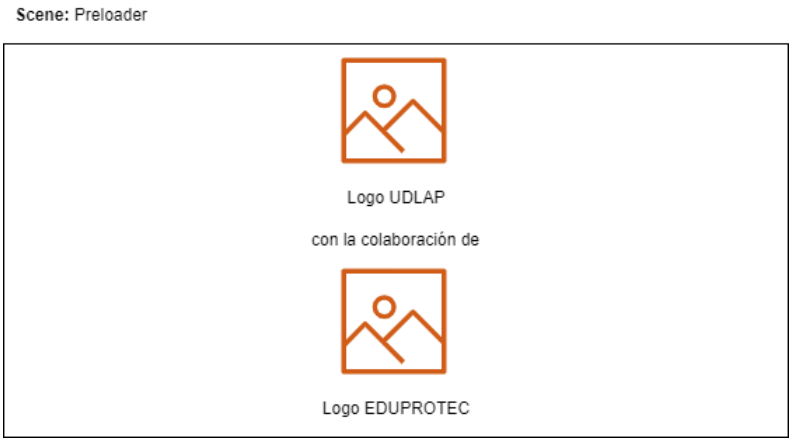


Figura 4. Ejemplo de una escena que se titula “Preloader”, la cual posee una imagen con el logo de la UDLAP, una etiqueta de texto que lee “con la colaboración de”, y una imagen con el logo de *EDUPROTEC*.

3.1.2. Íconos estandarizados

Para representar imágenes, animaciones, grabaciones de voz, y otros recursos audiovisuales que se encuentran en cada escena, se emplean los siguientes íconos estandarizados (véase Tabla 1):

Ícono	Elemento representado
	Imagen o imagen de fondo
	Grabación de voz
	Música
	Animación (escena)
	Animación (tutorial de juego)

Tabla 1. Íconos representativos de los elementos encontrados en una wireframe.

Al implementar el prototipo de la aplicación, dichos recursos audiovisuales se encuentran almacenados en directorios correspondientes a las escenas en la cual se utilizan; o, en caso de ser utilizados en múltiples escenas, en una carpeta compartida. Para acceder a estos elementos durante la ejecución de la aplicación (e.g., las imágenes de las frutas durante el juego de Conteo), se incluyen referencias a los mismos mediante los atributos de las clases que controlan cada escena. Los recursos estáticos, en contraste, como lo son las imágenes de fondo, los personajes, y todos aquellos cuya posición se mantiene fija o que no poseen interactividad con el usuario no son accedidos durante la ejecución de la aplicación.

3.1.3. Cuadro de Diálogo

En la aplicación existen varias escenas en las que los personajes se comunican entre sí, o con el usuario; estas conversaciones se representan mediante *cuadros de diálogo*. Un cuadro de diálogo es un panel horizontal que despliega una oración de una conversación a la vez (véase Figura 5); al mismo tiempo, se reproduce una grabación de voz que la lee en voz alta.

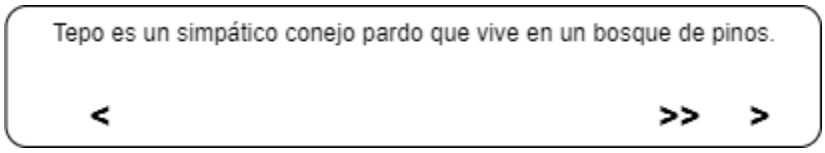


Figura 5. Ejemplo de un cuadro de diálogo, el cual presenta la oración “Tepo es un simpático conejo pardo que vive en un bosque de pinos”.

Adicionalmente, el cuadro de diálogo posee tres botones, los cuales controlan el flujo del diálogo como se indica en la siguiente Tabla 2.

Botón	Acción
<	Regresa al enunciado previo
>	Avanza al siguiente enunciado
>>	Omite las escenas con diálogo.

Tabla 2. Botones del cuadro de diálogo. Dependiendo del botón que se oprima, el diálogo regresará, avanzará, o se omitirá.

3.2. El diseño de las secciones desarrolladas en esta tesis

3.2.1. Sección: Introducción

La Introducción es un grupo de siete escenas en las que se le relata a nuevos usuarios la historia y el contexto del mundo de la aplicación, esto con el fin de ambientarlos y motivarlos a que se involucren y participen en sus juegos (véase Figura 6). Para ello se decidió extender la temática y los personajes de *Los Números del Bosque* a la aplicación completa, para no limitarlos exclusivamente al juego de Conteo.

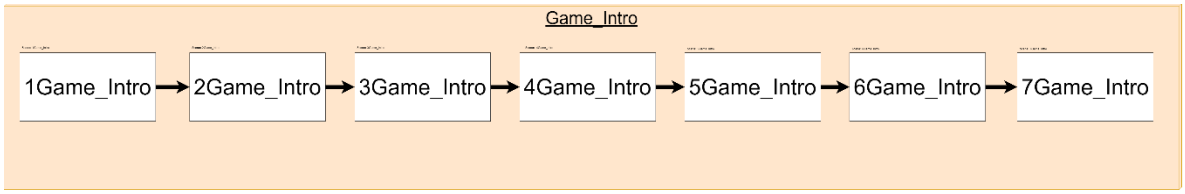


Figura 6. Wireframe de las escenas que conforman la sección de Introducción.

Las siete escenas que conforman la sección se titulan *1Game_Intro*, *2Game_Intro*, y así sucesivamente hasta *7Game_Intro* (véase Figuras 6A – 6G);

dichas escenas son, en esencia, cinemáticas, lo cual significa que transcurren en un orden lineal y consecutivo, son autodescriptivas, y no poseen mayor interactividad al que se incluye en los cuadros de diálogo, por lo que no requieren una explicación individual con el mismo detalle que las escenas de las otras secciones.

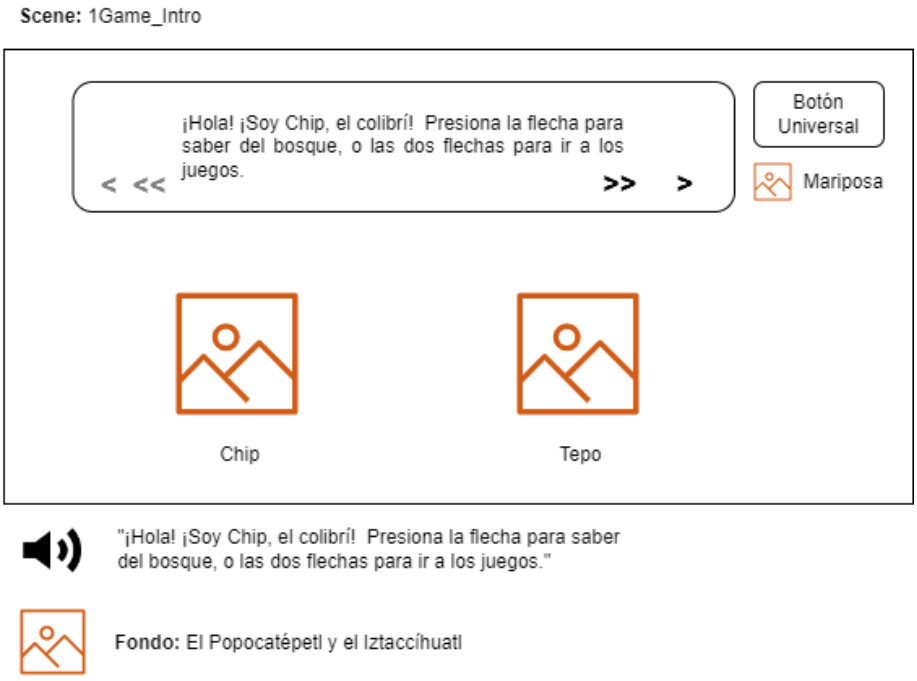
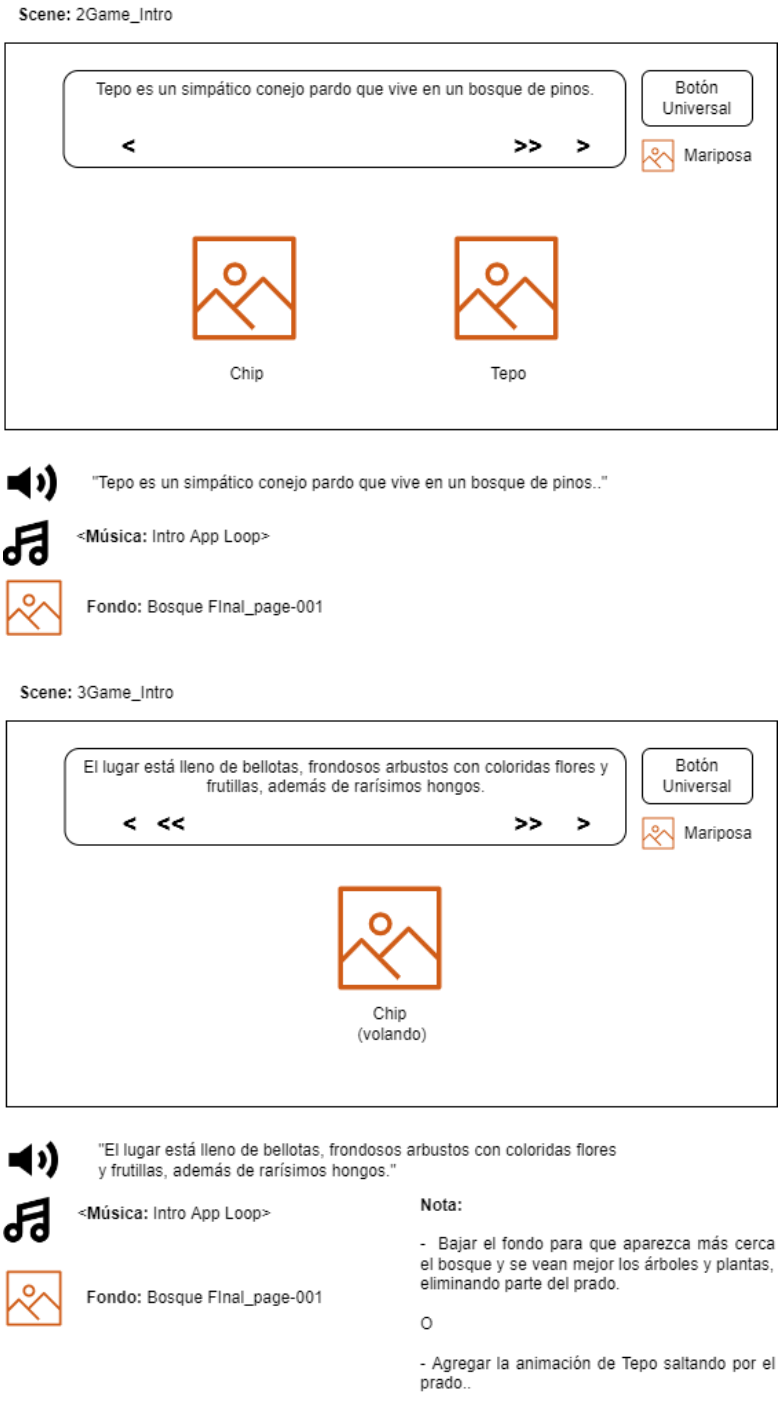
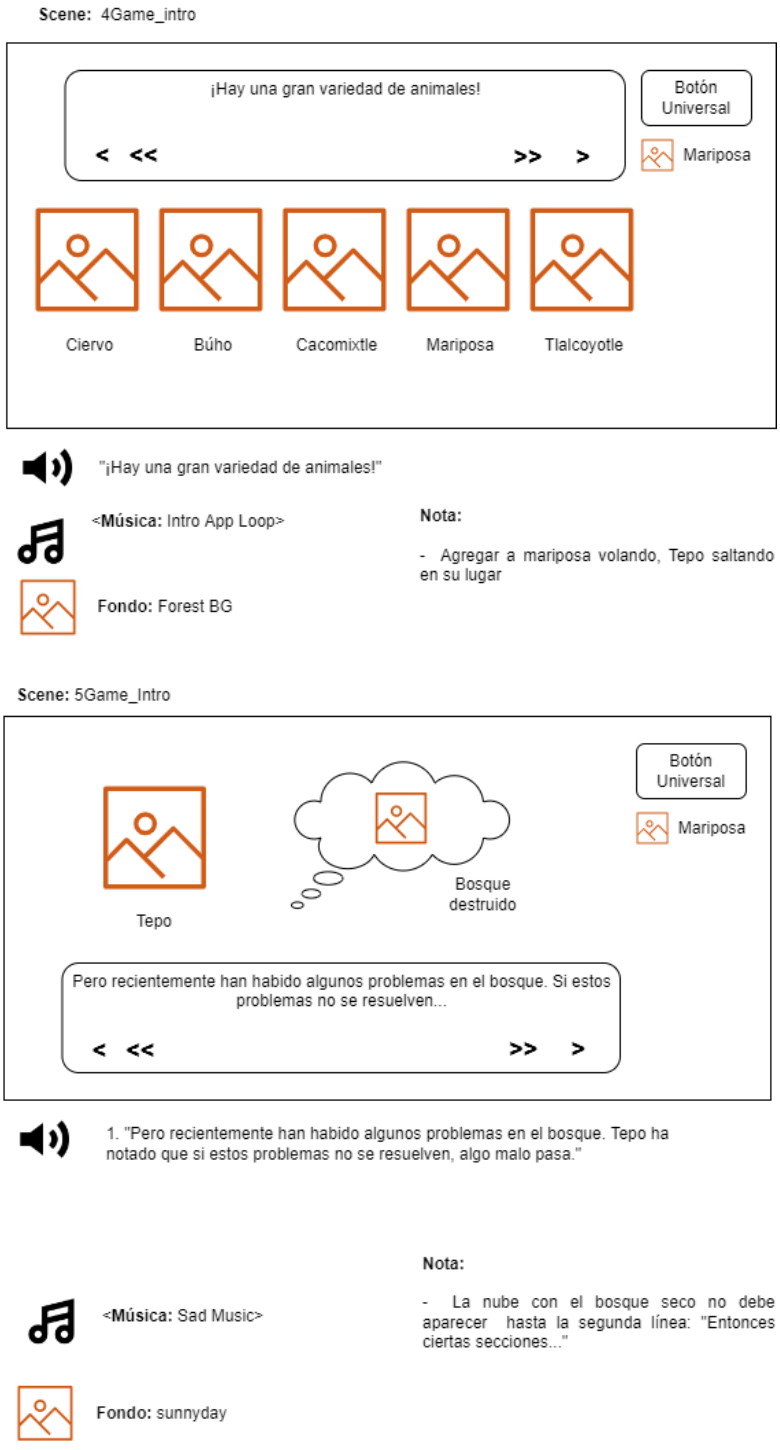


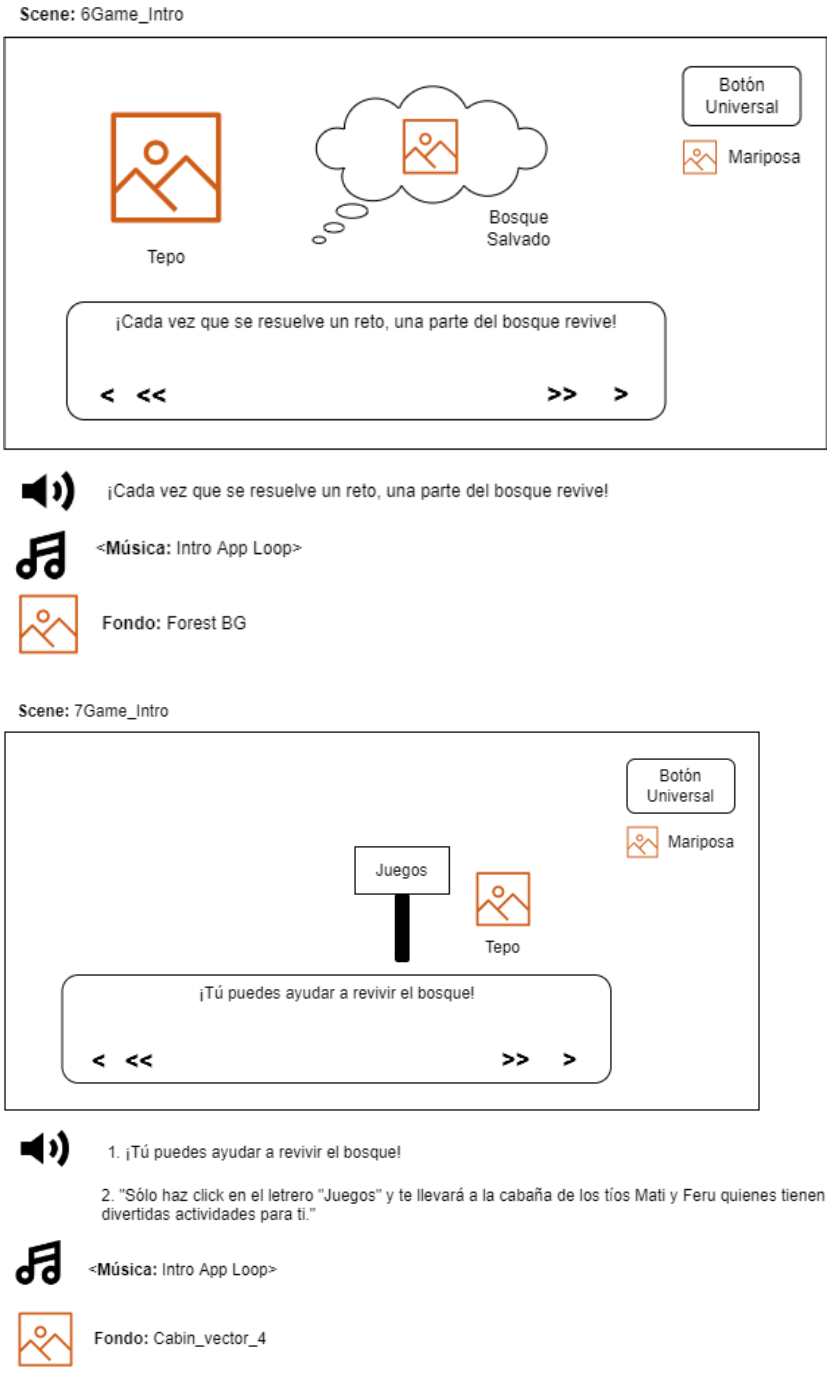
Figura 6A. Wireframe de la escena *1Game_Intro*, de la sección de Introducción.



Figuras 6B y 6C. Wireframe de las escenas *2Game_Intro* (superior) y *3Game_Intro* (inferior), de la sección de Introducción.



Figuras 6D y 6E. Wireframe de las escenas *4Game_Intro* (superior) y *5Game_Intro* (inferior), de la sección de Introducción.



Figuras 6F y 6G. Wireframe de las escenas *6Game_Intro* (superior) y *7Game_Intro* (inferior), de la sección de Introducción.

La única escena de la sección con interactividad adicional al cuadro de diálogo es *7Game_Intro*, en la cual, al oprimir el letrero, el usuario será trasladado al menú de juegos. A continuación, se resumen en la Tabla 3 las escenas que conforman la sección de Introducción, y sus oraciones de diálogo correspondientes:

Escena	Oraciones de diálogo
1Game_Intro	"¡Hola! ¡Soy Chip, el colibrí! Presiona la flecha (>) para saber del bosque, o las dos flechas (>>) para ir a los juegos."
2Game_Intro	"Tepo es un simpático conejo pardo que vive en un bosque de pinos."
3Game_Intro	"El lugar está lleno de bellotas, frondosos arbustos con coloridas flores y frutillas, además de rarísimos hongos."
4Game_Intro	"¡Hay una gran variedad de animales!"
5Game_Intro	1. "Pero recientemente han habido algunos problemas en el bosque. Tepo ha notado que si estos problemas no se resuelven, algo malo pasa." 2. "Entonces ciertas secciones del bosque se van volviendo grises, las plantas mueren y los animales dejan de vivir ahí"
6Game_Intro	"¡Cada vez que se resuelve un reto, una parte del bosque revive!"
7Game_Intro	1. "¡Tú puedes ayudar a revivir el bosque!" 2. "Sólo haz click en el letrero "Juegos" y te llevará a la cabaña de los tíos Mati y Feru quienes tienen divertidas actividades para ti"

Tabla 3. Las oraciones de la sección de Introducción, clasificadas por sus escenas correspondientes.

3.2.2. Sección: Conteo

El juego de Conteo es un grupo de siete escenas que adaptan *Los Números del Bosque*, en donde el usuario desarrolla sus habilidades matemáticas mediante el conteo de frutas (véase Figura 7).

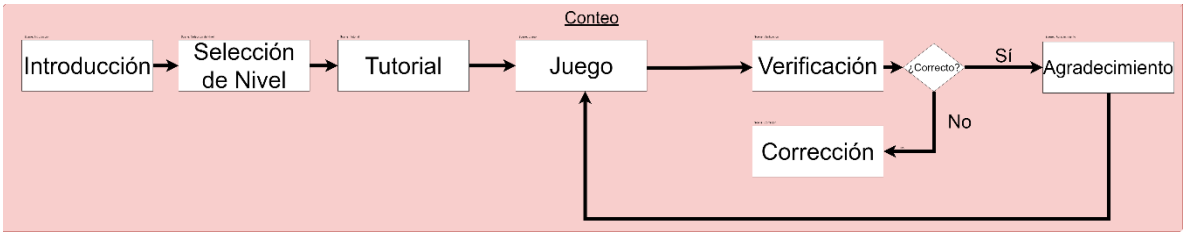


Figura 7. Wireframe de las escenas que conforman la sección de Conteo.

3.2.2.1. Escenas: *Introducción, Selección de Nivel, Tutorial*

Las primeras tres escenas del juego de Conteo (*Introducción*, *Selección de Nivel*, y *Tutorial*, respectivamente) son cinemáticas, cuya interactividad con el usuario se limita al cuadro de diálogo. *Introducción* y *Selección de Nivel* exponen el contexto del juego al usuario: los tíos Feru y Mati desean cocinar una tarta, por lo que le solicitan ayuda a Tepo para buscar las frutas que requieren (véase Figura 7A). Tepo, a su vez, le pide ayuda al jugador para recolectar las frutas encomendadas, y cerciorarse de que sean las correctas tanto en cantidad como en tipo (véase Figura 7B). Seguido, en *Tutorial*, se le instruye al usuario cómo jugar mediante el cuadro de diálogo y una animación demostrativa (véase Figura 7C).

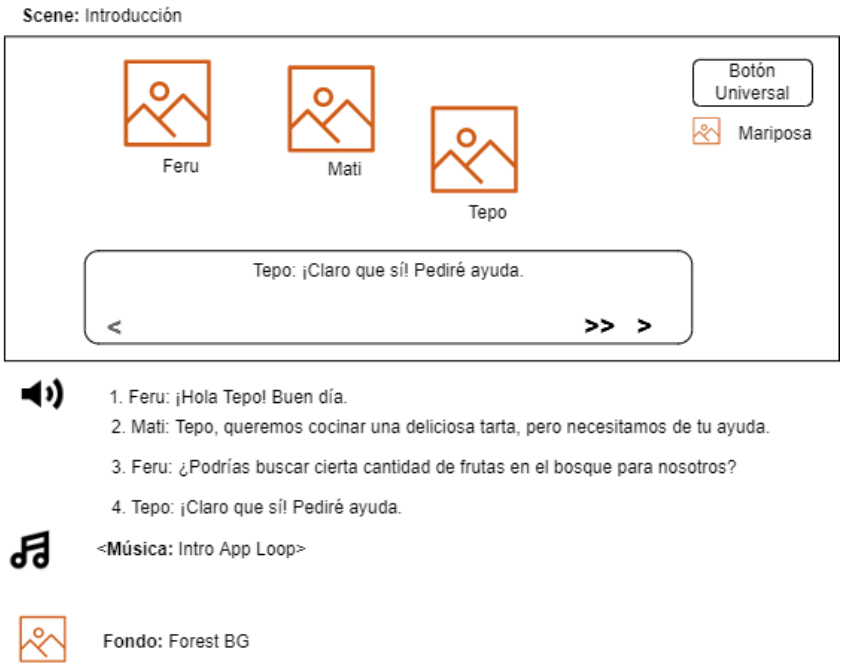


Figura 7A. Wireframe de la escena *Introducción*, de la sección de Conteo.

Figura 7B. Wireframe de la escena *Selección de Nivel*, de la sección de Conteo.

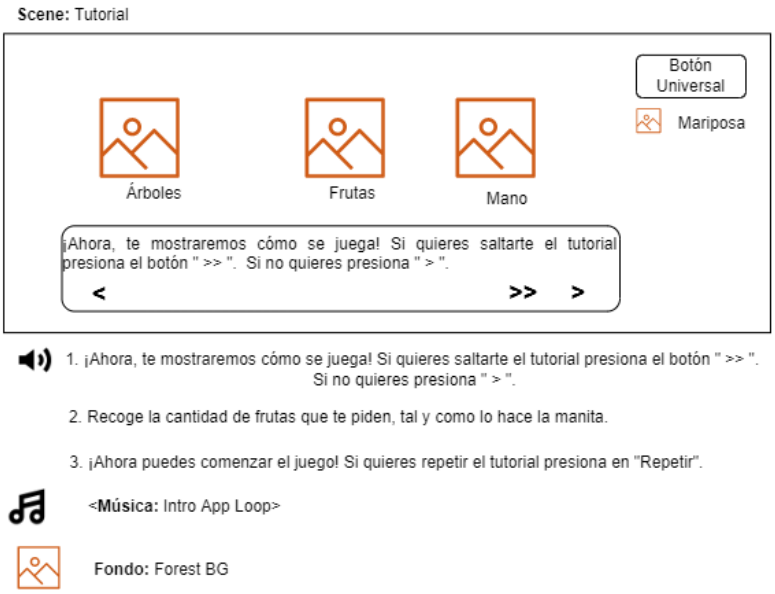


Figura 7C. Wireframe de la escena *Tutorial*, de la sección de Conteo.

En la siguiente Tabla 4 se enlistan las oraciones de diálogo correspondientes a cada escena de la sección de Conteo:

Escena	**Oraciones de diálogo**
Introducción	1. "Feru: ¡Hola Tepo! Buen día." 2. "Mati: Tepo, queremos cocinar una deliciosa tarta, pero necesitamos de tu ayuda" 3. "Feru: ¿Podrías buscar cierta cantidad de frutas en el bosque para nosotros?" 4. "Tepo: ¡Claro que sí! Pediré ayuda."
Selección de Nivel	1. "Los tíos Feru y Mati me han pedido ayuda para buscar por el bosque esta cantidad de frutas." 2. "Ayúdame para que puedan cocinar su deliciosa tarta."
Tutorial	1. "¡Ahora, te mostraremos cómo se juega! Si quieres saltarte el tutorial presiona el botón " >> ". Si no quieres presiona " > "". 2. "Recoge la cantidad de frutas que te piden, tal y como lo hace la manita." 3. "¡Ahora puedes comenzar el juego! Si quieres repetir el tutorial presiona en "Repetir"

Tabla 4. Las oraciones de las primeros tres escenas de la sección de Conteo, clasificadas por sus escenas correspondientes.

3.2.2.2. Escena: *Juego*

Después de concluir *Tutorial*, el usuario es trasladado a la escena *Juego*, donde podrá fomentar sus habilidades matemáticas mediante el conteo de frutas. La escena está compuesta por tres árboles de manzanas, peras, y duraznos, los cuales rodean una canasta de picnic en el centro. En la esquina superior izquierda se puede visualizar el número de frutas a recolectar de cada tipo, el cual se genera aleatoriamente al inicio del juego, tal que no se excedan de 10 frutas en total (véase Figura 8).

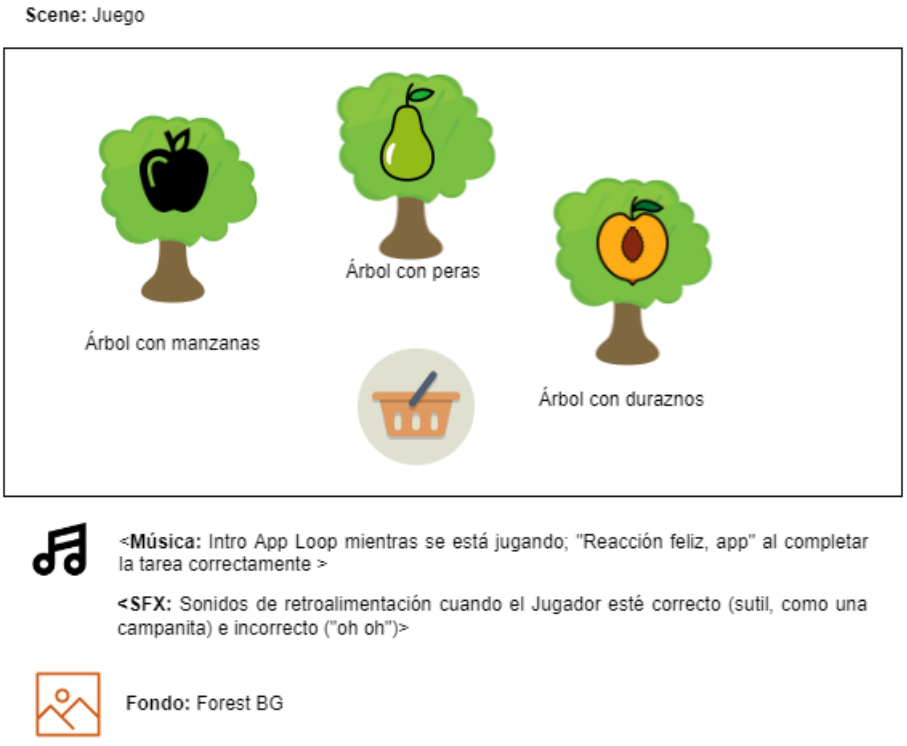


Figura 8. Diseño de la escena *Juego* de la sección de Conteo, en la cual el usuario desarrolla sus habilidades matemáticas.

El primer objetivo del juego de Conteo es arrastrar la cantidad indicada de frutas de los árboles a la canasta; cada vez que una fruta es arrastrada a la canasta, se incrementará un contador que indica el número de frutas recolectadas al momento, y se reproducirá un audio correspondiente a aquel número. Al terminar de arrastrar todas las frutas solicitadas exitosamente, el usuario será trasladado a la escena *Verificación* para confirmar su comprensión del juego.

3.2.2.3. Escena: *Verificación*

Al completar con éxito la escena *Juego*, el usuario es trasladado a la escena *Verificación*, donde deberá cumplir el segundo objetivo del juego de Conteo: demostrar su comprensión del concepto de número, su independencia de las frutas contadas, y su habilidad para agregarlas. La escena está integrada por un panel que contiene las frutas que se recolectaron en *Juego*, un cuadro de diálogo que le pregunta al usuario cuántas frutas fueron, y una slider con el cual indicar la respuesta (véase Figura 9).

Una vez que el usuario haya elegido su respuesta, se deberá oprimir el botón > del cuadro de diálogo para enviarla. Si su respuesta es acertada, el usuario es trasladado a la escena *Agradecimiento*; en caso contrario, será redirigido a la escena *Corrección*.

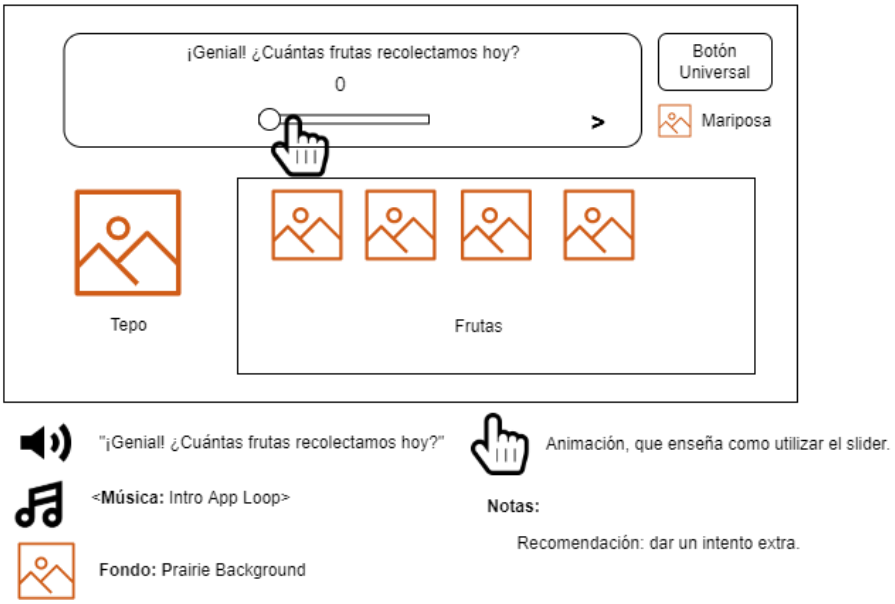


Figura 9. Diseño de la escena *Verificación* de la sección de Conteo, en la cual el usuario debe demostrar su comprensión del concepto de número.

3.2.2.4. Escena: *Corrección*

Si el usuario no consigue acertar en *Verificación* el número de frutas recolectadas en *Juego*, se le redirigirá a la escena *Corrección*, en donde los tíos Feru y Mati le explicarán cuál es la respuesta correcta y por qué. La escena está compuesta por un cuadro de diálogo, mediante el cual Feru y Mati tomarán turnos para presentarle al usuario la lista de frutas recolectadas por tipo (i.e., manzana, pera, durazno). Cada fruta se encuentra acompañada de un dígito que denota su posición en la lista, para que el usuario pueda contarlas y comprobar la respuesta de los tíos (véase Figura 10). Por último, se enlistarán todas las frutas, independientemente de su tipo, para contar el total de frutas recolectadas. Al terminar la explicación de los tíos Feru y Mati, se le preguntará al usuario si desea

seguir jugando, o si preferiría jugar algo más; dependiendo de la opción que elija, se le redirigirá a la escena *Juego*, o se le trasladará al menú de juegos, respectivamente.

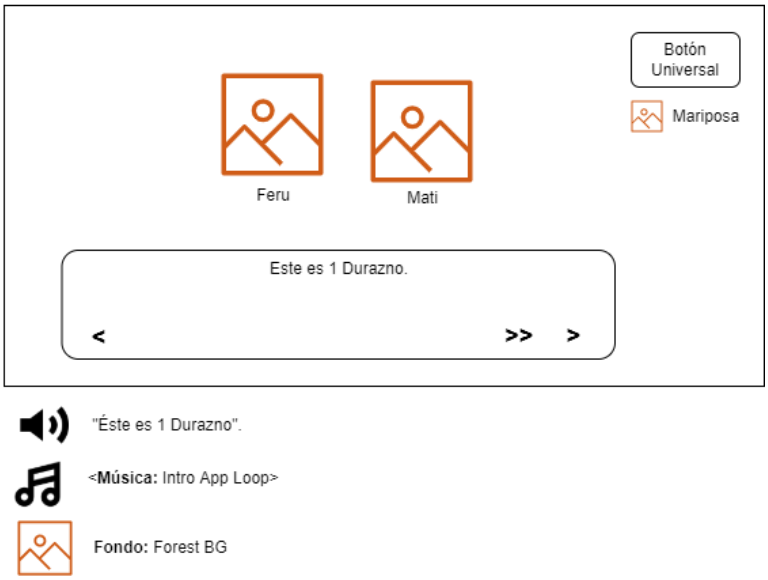


Figura 10. Diseño de la escena *Corrección* de la sección de Conteo, en la cual los tíos Feru y Mati le enseñan al usuario la respuesta correcta de *Verificación*.

3.2.2.5. Escena: *Agradecimiento*

Si el usuario acertó en *Verificación* el número de frutas recolectadas en *Juego*, se le redirigirá a la escena *Agradecimiento*, en donde se reproducirá una escena cinemática en la cual Chip y Tepo le agradecen al usuario por su ayuda, y le piden que vuelva pronto. Al ser cinemática, la escena consiste únicamente en un cuadro de diálogo y las imágenes de los personajes (véase Figura 11). Al concluir el agradecimiento de Chip y Tepo, se le preguntará al usuario si desea seguir

jugando, o si preferiría jugar algo más; dependiendo de la opción que elija, se le redirigirá a la escena *Juego*, o se le trasladará al menú de juegos, respectivamente.

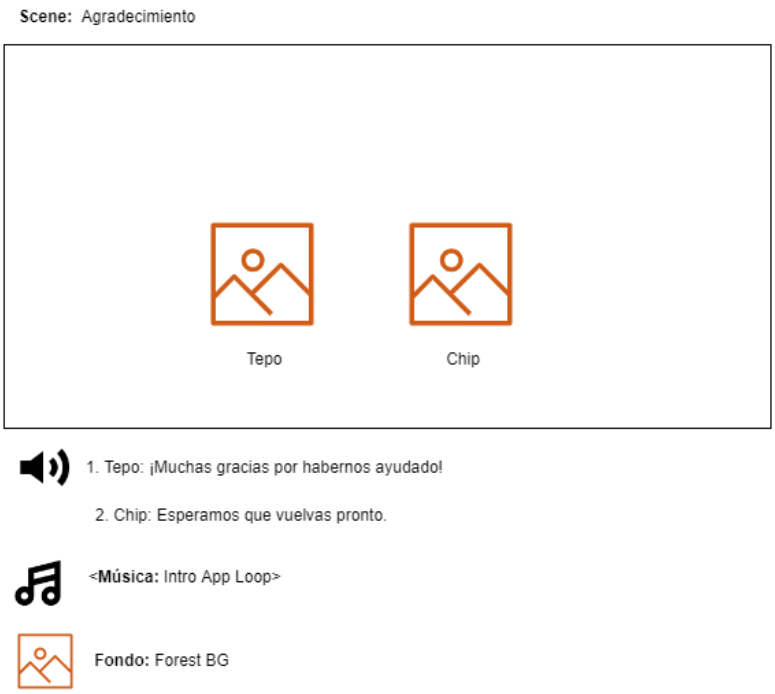


Figura 11. Diseño de la escena *Agradecimiento* de la sección de Conteo, en la cual Chip y Tepo le agradecen su participación al usuario.

3.2.3. Sección: Vocabulario

El juego de Vocabulario consiste en una escena, *Juego_Vocabulario*, en donde el usuario desarrolla sus habilidades de lectoescritura mediante el ordenamiento de letras y el relacionamiento de la palabra resultante con su significado. La escena está compuesta de una fila inferior de letras desordenadas, una fila superior de ranuras en donde embonan dichas letras, y un panel horizontal en el centro, en donde se despliega la definición de una palabra (véase Figura 12). El objetivo del juego es ordenar las letras provistas en la fila inferior, arrastrando cada una a su lugar

correspondiente de la fila superior, tal que la palabra resultante corresponda a la definición proporcionada en el panel central. Una letra arrastrada a una ranura solamente embonará si dicha letra se encuentra en la posición correspondiente de la palabra definida—es decir, el juego no permite la formación de una palabra que no corresponda al de la definición proporcionada. En caso de arrastrar una letra a una ranura que no le corresponde, la letra será regresada automáticamente a la fila inferior, haciéndole saber al usuario que la letra arrastrada no corresponde a esa posición de la palabra a formar.

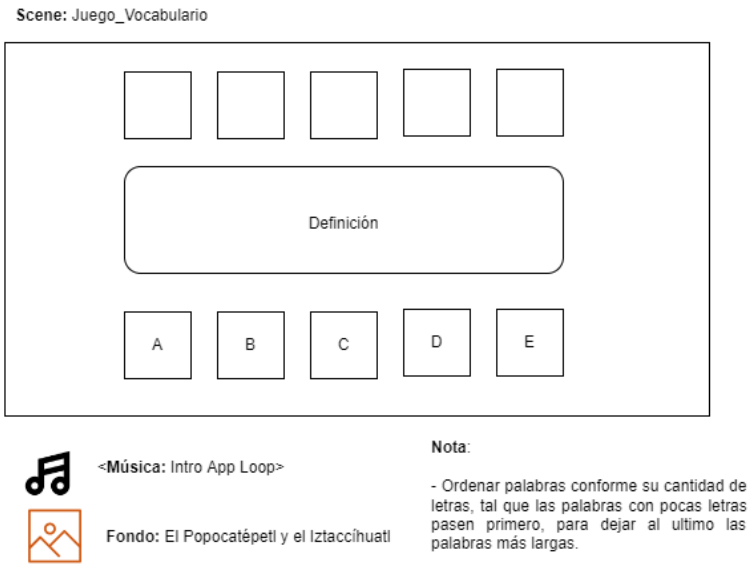


Figura 12. Diseño de la escena *Juego_Vocabulario*, única escena de la sección de Vocabulario, en la cual el usuario deberá reordenar las letras provistas en la fila inferior a su lugar correspondiente de la fila superior, tal que la palabra resultante se relacione a la definición proporcionada en el panel central.

4. Desarrollo del prototipo

Una vez diseñadas los wireframes, representativos de las secciones que entran en el alcance de esta tesis, se procedió a su implementación en el motor de videojuegos *Unity*.

4.1. Arquitectura del prototipo

El prototipo desarrollado sigue el patrón de diseño *Modelo-Vista-Controlador*, por lo que su arquitectura se divide en sus tres capas correspondientes de la siguiente manera.

4.1.1. *Modelo*

Debido a que las secciones que entran en el alcance de esta tesis (i.e., Introducción, juego de Conteo, y juego de Vocabulario) no almacenan información para su recuperación posterior, esta capa no se encuentra implementada en el prototipo. Sin embargo, se adoptó este patrón de diseño debido a que hay planes en el futuro para recaudar información del desempeño de cada jugador, almacenarla en una base de datos en la nube, y generar estadísticas, por lo que la implementación de esta capa se mantiene en consideración como trabajo a futuro.

4.1.2. *Vista*

Las interfaces gráficas con las que interactúa el usuario en cada sección son definidas en las escenas de *Unity*, por lo que éstas actúan como las vistas de la aplicación. Esta capa consiste en 15 escenas, las cuales han sido mencionadas en el capítulo anterior: *1Game_Intro, 2Game_Intro,* hasta *7Game_Intro* en la sección de Introducción; *Introducción*, *Selección de Nivel*, *Tutorial*, *Juego*, *Verificación*, *Corrección*, y

Agradecimiento en la sección de Conteo; y *Juego_Vocabulario* en la sección de Vocabulario.

4.1.3. *Controlador*

Unity posee una API de programación, la cual permite añadir funcionalidad a los recursos que ofrece mediante código escrito en el lenguaje C#, con el fin de que éstos se ajusten a los requisitos de la aplicación. Dicho código es responsable de percibir las acciones que realiza el usuario a través de las vistas, procesar sus datos y aplicar la lógica que estas acciones requieren, y actualizar la interfaz de usuario correspondientemente, por lo cual las clases de C# actúan como los controladores de la aplicación. Esta capa de la arquitectura está compuesta por las siguientes 10 clases de C#: *Dialog* en la sección de Introducción y Conteo; *Animations*, *FruitInitialization*, *FruitsToCollect*, *MatDragHandeler*, *Basket*, y *Monologue* en la sección de Conteo; y *VocabularyManager*, *DragHandeler*, y *Slot* en la sección de Vocabulario. Sus métodos y funcionalidad se detallan en el siguiente apartado.

4.2. Clases desarrolladas para los controladores

En los siguientes subapartados se definirán los atributos, métodos, y el propósito de las clases de C# que conforman los controladores del prototipo, clasificados por la sección en la cual se emplean. Un diagrama de clases de dichos controladores se puede apreciar en la siguiente Figura 13:

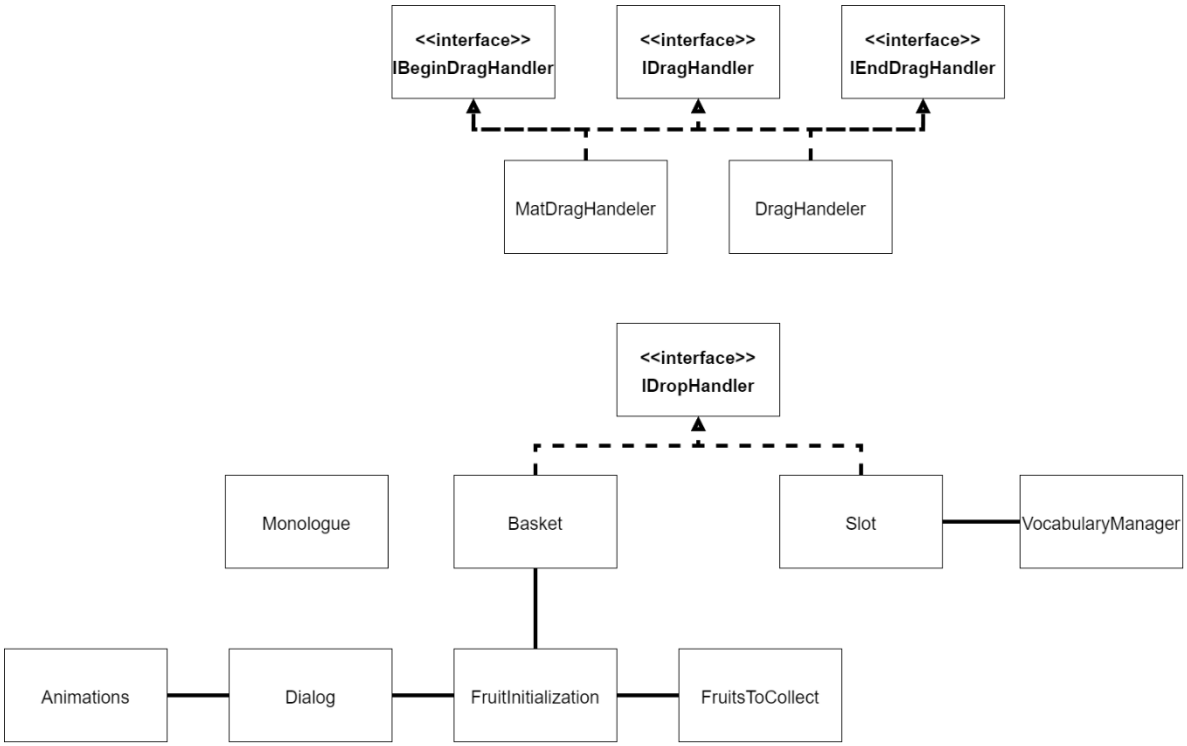


Figura 13. Diagrama de las clases desarrolladas para los controladores de la aplicación, incluyendo cuatro interfaces de la API de *Unity* que algunas clases emplean para complementar su funcionalidad.

4.2.1. Sección: Introducción

Debido a que la sección de Introducción, al ser de naturaleza cinemática, no posee mayor funcionalidad al que se incluye en el cuadro de diálogo, la única clase que se emplea en la misma es *Dialog*, la cual define su ejecución. *Dialog* posee como atributos un arreglo de las oraciones a escribir en cada escena, un arreglo de sus grabaciones de voz correspondientes, un índice que determina qué oración debe ser desplegada y reproducida en el momento presente, y referencias a los elementos de la escena y a la clase *FruitInitialization*, cuya operación se abordará en un subapartado posterior. Asimismo, contiene cuatro métodos de interés: *Type*,

NextSentence, *PrevSentence*, y *notstart*, cuyas funcionalidades varían ligeramente dependiendo de la escena en la que se encuentre el usuario.

Type es el método responsable de presentar las oraciones de una conversación de manera escrita y verbal. Cuando el método es invocado, la oración que indique el índice se presentará mediante el cuadro de diálogo de manera progresiva (i.e., letra por letra, a una velocidad determinada) y se reproducirá su grabación de voz correspondiente. Cuando el usuario se encuentra en la escena *Corrección* de la sección de Conteo, el método se apoya de la clase *FruitInitialization* para determinar cuáles fueron las frutas que le fueron solicitadas en la escena *Juego* con el fin de incluir sus cantidades, tipos, e imágenes en cada oración del cuadro de diálogo, y reproducir sus grabaciones acordemente. Una vez concluido el despliegue de una oración, se habilitarán los botones del cuadro de diálogo para otorgarle control al usuario de cómo debe proseguir el flujo de la conversación.

NextSentence es el método encargado de avanzar la conversación de una escena. Se encuentra vinculado al botón > del cuadro de diálogo, por lo que se invocará cada vez que éste se oprima. Cuando *NextSentence* es invocado, se incrementará el índice de la clase, así estableciendo que la oración consecutiva a la actual será la siguiente en ser desplegada. Seguido, se realizará una llamada al método *Type* para desplegar dicha oración mediante el cuadro de diálogo y reproducir su grabación de voz. Tras terminar de presentar todas las oraciones de una escena, *NextSentence* cargará la siguiente escena de su sección, a excepción de que el usuario se encuentre en las escenas *Corrección* y *Agradecimiento* de la sección de Conteo.

En tales casos, el método presentará un panel en donde se le preguntará al usuario si desea seguir jugando el juego de Conteo, o si preferiría jugar algo más; dependiendo de la opción que elija, se le redirigirá a la escena *Juego*, o se le trasladará al menú de juegos, respectivamente.

PrevSentence, en contraste con el método anterior, es el método responsable de retroceder la conversación de una escena. Se encuentra vinculado al botón **<** del cuadro de diálogo, por lo que se invocará cada vez que éste se oprima. Cuando *PrevSentence* es invocado, se decrementará el índice de la clase, así estableciendo que la oración anterior a la actual será la siguiente en ser desplegada. Seguido, se realizará una llamada al método *Type* para desplegar dicha oración mediante el cuadro de diálogo y reproducir su grabación de voz. Si el método se invoca al inicio de una conversación, se cargará la escena anterior, siempre y cuando el usuario no se encuentre en la primera escena de una sección; en tal caso, el botón se encontrará deshabilitado, impidiendo la invocación de *PrevSentence*.

notstart es el último método de interés de la clase *Dialog*, el cual es responsable de omitir las escenas con diálogo en una sección. Se encuentra vinculado al botón **>>** del cuadro de diálogo, por lo que se invocará cada vez que éste se oprima. Al ser invocado, el método cargará la primera escena de la sección que carezca de cuadro de diálogo. Es decir, cuando el método se ejecuta en la sección de Introducción, el usuario será redirigido al menú de juegos; cuando se ejecuta en la sección de Conteo, a la escena *Juego*.

Las clases que conforman la sección de Introducción, sus métodos, íconos asociados, y sus descripciones se resumen en la siguiente Tabla 5.

Clase	Método	Ícono asociado	Descripción
Dialog	Type	N/A	Presenta las oraciones de una conversación de manera escrita y verbal
	NextSentence	>	Avanza la conversación de una escena
	PrevSentence	<	Retrocede la conversación de una escena
	notstart	>>	Omite las escenas con diálogo en una sección

Tabla 5. Resumen de las clases y métodos presentes en la sección de Introducción, especificando sus íconos y descripción.

4.2.2. Sección: Conteo

En las siguientes subsecciones se describirán las características de las clases de C# que conforman los controladores de la sección de Conteo.

4.2.2.1. Clase: *Animations*

Animations es la clase responsable de habilitar y deshabilitar las animaciones de los personajes cuando conversan mediante el cuadro de diálogo. La clase posee como atributos un arreglo de las animaciones de los personajes que conversan en una escena, la escena en la que se encuentra el usuario, y una referencia a la clase

Dialog, la cual se emplea para obtener la oración que será desplegada en el cuadro de diálogo. La clase no posee métodos a excepción de aquellos que son generados automáticamente por *Unity*, por lo que su funcionalidad se encuentra definida en el método *Update*, mismo que se invoca cada fotograma de la escena. Dependiendo de la oración que se encuentra siendo desplegada, se habilitará la animación del personaje que la comunica. Una vez que dicha oración haya concluido, la animación del personaje correspondiente se deshabilitará.

4.2.2.2. Clase: *FruitInitialization*

FruitInitialization es la clase encargada de establecer la cantidad de frutas, el orden en que le serán solicitadas al usuario, y de crearlas en las escenas requeridas de la sección. La clase posee como atributos un arreglo de los tipos de frutas que dispone el juego (i.e., manzana, pera, durazno) y un arreglo de sus cantidades solicitadas. Los valores iniciales de dichos atributos son otorgados y empleados por dos métodos de interés: *InitializeFruits*, el cual se invoca en la escena *Juego*, e *InstantiateFruits*, que se ejecuta en las escenas *Juego*, *Verificación*, y *Corrección*. Ambos métodos se definen a continuación.

InitializeFruits es el método responsable de asignar la cantidad de frutas que le serán solicitadas al usuario, y a qué árbol corresponderá cada tipo de fruta. Al ser invocado, el método genera cantidades aleatorias para cada tipo de fruta, tal que cada una sea solicitada al menos una vez y la cantidad total de frutas solicitadas no se exceda de 10. Asimismo, la vinculación entre cada árbol y el tipo de fruta que contienen también es generado de manera aleatoria.

InstantiateFruits es el método encargado de crear las frutas asignadas en el método anterior. Cuando es ejecutado en la escena *Juego*, el método creará para cada árbol la cantidad idéntica de frutas, equivalente a la cantidad de frutas mayor solicitada. Por ejemplo, si el juego le solicita al usuario recolectar 1 manzana, 4 peras, y 5 duraznos, se crearán 5 manzanas, peras, y duraznos en sus árboles correspondientes porque la cantidad de frutas mayor solicitada es 5. En la escena *Verificación*, el método desplegará mediante un panel las frutas que fueron recolectadas en la escena *Juego*, y en la escena *Corrección* se mostrarán mediante el cuadro de diálogo las frutas que le fueron solicitadas al usuario en la escena *Juego*.

4.2.2.3. Clase: *FruitsToCollect*

FruitsToCollect es la clase responsable de instruirle al usuario las frutas a recolectar en la escena *Juego*. La clase posee como atributo una referencia a la clase *FruitInitialization*, la cual se emplea para conocer la cantidad solicitada de cada tipo de frutas. *FruitsToCollect* no posee métodos a excepción de aquellos que son generados automáticamente por *Unity*, por lo que su funcionalidad se encuentra definida en el método *Start*, mismo que se invoca al inicio de la escena. Las instrucciones del juego son generadas dinámicamente utilizando la referencia a la clase *FruitInitialization*, especificando para cada tipo de fruta el número solicitado a recolectar seguido de una imagen de la misma.

4.2.2.4. Clase: *MatDragHandeler*

MatDragHandeler es la clase encargada de habilitar el arrastre de las frutas en la escena *Juego*. Para ello, la clase implementa tres interfaces de la API de *Unity*: *IBeginDragHandler*, *IDragHandler*, e *IEndDragHandler*, las cuales proveen métodos para integrar funcionalidad *drag and drop* en la aplicación. *MatDragHandeler* posee como atributos la fruta que está siendo arrastrada por el usuario, y su posición inicial; dichos atributos son manipulados por tres métodos de interés: *OnBeginDrag*, *OnDrag*, y *OnEndDrag*, cuyas firmas son proporcionadas por las interfaces implementadas. Estos métodos representan la lógica a ejecutar antes, durante, y después de arrastrar una fruta, respectivamente.

OnBeginDrag es el método invocado antes del arrastre de una fruta. Al ejecutarse, el método mantiene registro de la posición inicial de la fruta, y evita que ésta registre colisiones con otros elementos de la escena. *OnDrag* es el método invocado mientras una fruta se encuentra siendo arrastrada. Al ejecutarse, la posición de la fruta cambia de acuerdo a la posición del cursor/dedo del usuario. *OnEndDrag* es el método invocado al concluir el arrastre de una fruta. Al ejecutarse, se habilita el registro de colisiones entre la fruta con los otros elementos de la escena. En caso de que la posición en la que se haya dejado la fruta no corresponde con la canasta en la que se necesita depositar, la fruta será regresada automáticamente a su posición inicial.

4.2.2.5. Clase: *Basket*

Basket es la clase responsable de registrar cuando una fruta ha sido arrastrada a

la canasta de picnic en la escena *Juego*, así como de contar cuántas frutas han sido arrastradas hasta el momento. Para registrar cuando una fruta haya sido soltada en la canasta, la clase implementa la interfaz *IDropHandler* de la API de *Unity*. *Basket* posee como atributos un contador de las frutas depositadas en la canasta hasta el momento, una etiqueta de texto que despliega dicho contador en la escena, un arreglo de contadores para cada tipo de fruta recibida, una referencia a la clase *FruitInitialization* para conocer la cantidad solicitada de cada tipo de frutas, y un arreglo de diez audios, cada uno contando un número del uno al diez. Estos atributos son empleados por dos métodos de interés: *OnDrop* y *SwitchScene*.

OnDrop es el método encargado de registrar cuando una fruta haya sido arrastrada a la canasta, y de incrementar el contador de frutas recolectadas. El método es provisto por la interfaz *IDropHandler*, misma que es implementada por la clase, y se invoca cuando la fruta haya sido soltada sobre la canasta. Al ejecutarse, el método destruye la fruta arrastrada e incrementa el contador de las frutas recolectadas. Seguido, el método modifica la etiqueta de texto para reflejar dicho contador, y reproduce por audio el número de frutas que se han recolectado hasta el momento. Asimismo, dependiendo del tipo de fruta que haya sido arrastrada, se incrementa su contador correspondiente. Cuando el contador de frutas recolectadas equivale a la cantidad de frutas solicitadas, dicha cantidad se le informa a la clase *Monologue* (cuyo funcionamiento se explicará en el siguiente apartado), y se invoca el método *SwitchScene*.

SwitchScene es el método responsable de verificar que las frutas recolectadas corresponden a las frutas que le fueron solicitadas al usuario; de ser así, el método traslada al usuario a la escena *Verificación* para continuar con la siguiente etapa el juego. En caso contrario—es decir, el usuario recolectó la *cantidad* correcta de frutas, mas dichas frutas no corresponden a las que le fueron solicitadas—, el método le indicará a la clase *FruitInitialization* que el usuario se equivocó, y por ende no se le puede asignar nuevas frutas en la escena *Juego* hasta que acierte. Por último, se le trasladará a la escena *Corrección*.

4.2.2.6. Clase: *Monologue*

Monologue es la clase encargada de presentarle al usuario en la escena *Verificación* las frutas que recolectó en la escena *Juego*, y probar su entendimiento del mismo mediante el conteo de las frutas. La clase posee como atributos la cantidad de frutas recolectadas (misma que fue provista por la clase *Basket*), el valor que representa la slider de la escena, una grabación de voz, y referencias a los elementos de la escena. Estos atributos son utilizados por dos métodos de interés: *Type* y *Verify*.

Type es el método responsable de preguntarle al usuario cuántas frutas recolectó en la escena *Juego*, y habilitar el uso de la slider que indique su respuesta. Cuando el método es invocado al inicio de la escena *Verificación*, la oración "¡Genial! ¿Cuántas frutas recolectamos hoy?" se presentará mediante el cuadro de diálogo de manera progresiva (i.e., letra por letra, a una velocidad determinada) y se reproducirá su grabación de voz correspondiente. Al terminar

de desplegar la oración, se habilitarán la slider de la escena y una etiqueta de texto que muestra el valor de la anterior.

Verify es el método encargado de evaluar la respuesta del usuario, y de trasladarlo a la escena correspondiente dependiendo de si es correcta o no. Se encuentra vinculado al botón > del cuadro de diálogo de la escena *Verificación*, por lo que se invocará cada vez que éste se oprima. Al ser invocado, *Verify* comparará el valor de la slider con la cantidad de frutas recolectadas en la escena *Juego*; si son equivalentes, el método le indicará a la clase *FruitInitialization* que el usuario acertó su respuesta, y por ende ésta le puede asignar nuevas frutas la próxima vez que juegue, y lo trasladará a la escena *Agradecimiento*. En caso contrario, el método le indicará a *FruitInitialization* que el usuario se equivocó, y por ende no se le puede asignar nuevas frutas en la escena *Juego* hasta que acierte, y lo trasladará a la escena *Corrección*.

Las clases que conforman la sección de Conteo, sus métodos, y sus descripciones se resumen en la siguiente Tabla 6.

Clase	Método	Descripción
Animations	Update	Habilita la animación del personaje que comunica una oración
FruitInitialization	InitializeFruits	Asigna la cantidad de frutas que le serán solicitadas al usuario
	InstantiateFruits	Crea las frutas asignadas en el método anterior.
FruitsToCollect	Start	Omite las escenas con diálogo en una sección
MatDragHandeler	OnBeginDrag	Mantiene registro de la posición inicial de la fruta
	OnDrag	Cambia posición de la fruta de acuerdo a la posición del cursor/dedo del usuario
	OnEndDrag	Habilita el registro de colisiones entre la fruta con otros elementos de la escena
Basket	OnDrop	Registra cuando una fruta haya sido arrastrada a la canasta
	SwitchScene	Verifica que las frutas recolectadas correspondan a las frutas que solicitadas
Monologue	Type	Pregunta al usuario cuántas frutas recolectó en la escena *Juego*
	Verify	Evalúa la respuesta del usuario mediante el botón > del cuadro de diálogo

Tabla 6. Resumen de las clases y métodos presentes en la sección de Conteo.

4.2.3. Sección: Vocabulario

En las siguientes subsecciones se describirán las características de las clases de C# que conforman los controladores de la sección de Vocabulario.

4.2.3.1. Clase: *VocabularyManager*

VocabularyManager es la clase responsable de preparar el juego de Vocabulario en la escena *Juego_Vocabulario*, La clase posee como atributos un diccionario de las palabras de las que consiste el juego, cada una relacionada a su definición correspondiente, una pila que almacena las palabras que ya transcurrieron en el juego, un par de audios (i.e., uno cuando el usuario acierta, embonando una letra en una ranura, y otro cuando el usuario se equivoca), y referencias a los elementos de la escena. Dichos atributos se emplean por tres métodos de interés: *BuildGame*, *InstanceGameObjects*, y *EndWord*, cuyo funcionamiento se define a continuación.

BuildGame es el método encargado de aumentar el nivel del juego cuando sus tres palabras han transcurrido. Cuando es invocado al inicio de la escena, el método verifica si todas las palabras del nivel se encuentran en la pila de palabras transcurridas; de ser así, aumenta el nivel y recupera las tres palabras que lo conforman. Seguido, *BuildGame* invoca al método *InstanceGameObjects*, independientemente de si el nivel aumentó o no.

InstanceGameObjects es el método responsable de crear los elementos de los cuales consiste el nivel. Para ello, el método recupera una palabra aleatoria del

nivel, siempre y cuando no haya transcurrido; seguido, dicha palabra se coloca en la pila de palabras transcurridas para evitar repetir palabras en el juego. El método crea y adjunta cada una de sus letras en la fila inferior de la escena, así como una ranura correspondiente en la fila superior. A continuación, las letras de la fila inferior se reordenan, evitando que sus nuevas posiciones correspondan con sus posiciones originales en la palabra. Por último, se obtiene la definición correspondiente a la palabra recuperada, la cual se despliega mediante el panel central de la escena.

EndWord es el método encargado de limpiar la escena cuando el usuario haya concluido de reordenar las letras de la palabra, tal que coincidan con la definición proporcionada. Al invocarse, todas las letras y ranuras de la escena se destruyen, la definición se elimina del panel, y se invoca el método *BuildGame* para recuperar la siguiente palabra del juego.

4.2.3.2. Clase: *DragHandeler*

DragHandeler es la clase encargada de habilitar el arrastre de las letras en la escena *Juego_Vocabulario*. Para ello, la clase implementa tres interfaces de la API de *Unity*: *IBeginDragHandler*, *IDragHandler*, e *IEndDragHandler*, las cuales proveen métodos para integrar funcionalidad *drag and drop* en la aplicación. *DragHandeler* posee como atributos la letra que está siendo arrastrada por el usuario, y su posición inicial; dichos atributos son manipulados por tres métodos de interés: *OnBeginDrag*, *OnDrag*, y *OnEndDrag*, cuyas firmas son

proporcionadas por las interfaces implementadas. Estos métodos representan la lógica a ejecutar antes, durante, y después de arrastrar una fruta, respectivamente.

OnBeginDrag es el método invocado antes del arrastre de una letra. Al ejecutarse, el método mantiene registro de la posición inicial de la letra, y evita que ésta registre colisiones con las ranuras de la escena. *OnDrag* es el método invocado mientras una letra se encuentra siendo arrastrada. Al ejecutarse, la posición de la letra cambia de acuerdo a la posición del cursor/dedo del usuario. *OnEndDrag* es el método invocado al concluir el arrastre de una letra. Al ejecutarse, se habilita el registro de colisiones entre la letra con las ranuras de la escena. En caso de que la posición en la que se haya dejado la letra no corresponde con una ranura en la que se necesita depositar, la letra será regresada automáticamente a su posición inicial.

4.2.3.3. Clase: *Slot*

Slot es la clase responsable de registrar cuando una letra ha sido arrastrada a una ranura de la escena *Juego_Vocabulario*, así como de verificar si dicha letra corresponde a tal ranura. Para registrar cuando una letra haya sido soltada en una ranura, la clase implementa la interfaz *IDropHandler* de la API de *Unity*. *Slot* posee como atributos la letra que tiene permitida embonar, un contador de las letras que se han embonado hasta el momento, y una referencia a la clase *VocabularyManager* para hacer uso de sus atributos y métodos. Estos atributos son empleados por dos métodos de interés: *OnDrop* e *IncorrectSlotCoroutine*.

OnDrop es el método encargado de registrar cuando una letra haya sido arrastrada a una ranura, y de incrementar el contador de letras embonadas. El método es provisto por la interfaz *IDropHandler*, misma que es implementada por la clase, y se invoca cuando la letra haya sido soltada sobre una ranura. Al ejecutarse, el método verifica si la ranura se encuentra vacía (en otras palabras, sin una letra embonada) y si la letra arrastrada corresponde con aquella que la ranura permite embonar. En tal caso, la letra se embona en la ranura, se tiñe de color verde, y se reproduce el audio de acierto provisto por la clase *VocabularyManager*. Asimismo, el contador de letras embonadas incrementa con cada acierto del usuario; cuando éste equivale al número de ranuras—es decir, cuando el usuario haya completado de arrastrar todas las letras a sus ranuras correspondientes—, el contador de letras se reinicia, y se manda a llamar al método *EndWord* de la clase *VocabularyManager* para recuperar la siguiente palabra del juego. En el caso de que el usuario arrastre una letra a una ranura que no le corresponda, dicha letra se teñirá de amarillo y se invocará el método *IncorrectSlotCoroutine*.

IncorrectSlotCoroutine es el método responsable de notificarle al usuario que la letra que intentó embonar no corresponde a la ranura a la que se arrastró. Para ello, el método reproduce el audio de equivocación provisto por la clase *VocabularyManager*, la letra regresará a su posición original de la fila inferior, y después de un segundo, su tinte regresará a su color original.

Las clases que conforman la sección de Vocabulario, sus métodos, y sus descripciones se resumen en la siguiente Tabla 7.

Clase	Método	Descripción
VocabularyManager	BuildGame	Aumentar el nivel del juego cuando sus tres palabras han transcurrido
	InstanceGameObjects	Crea las ranuras y letras del nivel.
	EndWord	Limpia la escena cuando el usuario haya concluido de reordenar las letras de la palabra
DragHandeler	OnBeginDrag	Mantiene registro de la posición inicial de la letra
	OnDrag	Cambia posición de la letra de acuerdo a la posición del cursor/dedo del usuario
	OnEndDrag	Habilita el registro de colisiones entre la letra con las ranuras de la escena
Slot	OnDrop	Registra cuando una letra haya sido arrastrada a una ranura
	IncorrectSlotCoroutine	Notifica al usuario que la letra que intentó embonar no corresponde a la ranura a la que se arrastró

Tabla 7. Resumen de las clases y métodos presentes en la sección de Vocabulario.

El proyecto es de código abierto y se encuentra en un repositorio público de *GitHub*, el cual se puede consultar a través del siguiente vínculo: https://github.com/OCervantes/Sonador-Libros

5. Evaluación de prototipo

Habiendo desarrollado un prototipo funcional de la aplicación, se procedió a su evaluación por parte de la Dra. Zobeida Jezabel Guzmán Zavaleta, la Dra. Laura Helena Porras Hernández, Thelma Calderón Mayo, estudiante de la licenciatura en Pedagogía e integrante del mismo equipo, así como ocho alumnos de la clase *Interacción Humano-Computadora* (LIS3032) durante el periodo Otoño 2020.

5.1. Escenas resultantes

A continuación, se muestran las versiones iniciales de un par de las escenas de las cuales consiste la aplicación; en nombre, las escenas *Juego* de la sección de Conteo (véase Figura 14), y la escena *Juego_Vocabulario* de la sección de Vocabulario (véase Figura 15), respectivamente.

Figura 14. Implementación inicial de la escena *Juego* de la sección de Conteo.

Figura 15. Implementación inicial de la escena *Juego_Vocabulario* de la sección de Vocabulario.

5.2. Evaluaciones

Con base en el prototipo desarrollado, se realizaron tres evaluaciones: una técnica, ejecutada por la Dra. Guzmán (véase Anexo 2), una pedagógica puesta en práctica por la Dra. Porras (véase Anexo 3), y un *focus group* realizada entre ocho estudiantes de *Interacción Humano-Computadora* (LIS3032) durante el periodo Otoño 2020. Todas las evaluaciones fueron remotas debido al confinamiento de la pandemia, tal que las primeras dos fueron llevadas a cabo sin supervisión del equipo de desarrollo, por lo que cada evaluadora pudo examinar la aplicación a su parecer, sin guía u orientación, lo cual proveyó un ambiente similar al de un usuario nuevo que se encuentra utilizando la aplicación por primera vez. El *focus group*, en contraste, se efectuó en presencia de dos miembros del equipo de desarrollo, quienes realizaron una demostración del prototipo frente a la clase y, seguido, se condujo una discusión grupal de su usabilidad. Los resultados de la evaluación técnica fueron clasificadas en las pantallas de la aplicación, sus botones, funcionalidad, audio, así como retroalimentación de qué remover, agregar,

y cambiar. En contraste, las observaciones de la evaluación pedagógica se concentraron en los aciertos, cambios, y sugerencias para cada pantalla. Los resultados de antedichas evaluaciones se detallan a continuación:

5.2.1. Técnica

Entre las áreas de oportunidad observadas, el principal problema reportado recae en la carencia de responsividad de la aplicación—esto es, la habilidad de adaptar los elementos de la interfaz de usuario a las distintas resoluciones que se pueden presentar en dispositivos móviles—, por lo que en la escena *Juego* de la sección de Conteo se pueden desacomodar las frutas de cada árbol, ya sea apareciendo fuera del rango de su árbol, u ocultándose detrás de uno tal que resulte imposible arrastrarlos a la canasta. Asimismo, las grabaciones de voz reproducidas al momento en que una fruta es arrastrada a la canasta fueron reportada como ruidosas, confusas, y no funcionales para el conteo de cada fruta, por lo cual se sugirió reemplazarlas por grabaciones de voz que cuenten las cantidades de frutas por cada tipo (e.g., una manzana, dos manzanas, una pera, tres manzanas, etc.).

Un cambio mayor que se propuso es la inclusión de un botón que indique cuando el usuario se sienta listo para avanzar, independientemente de si esté correcto o no, en contraste con el modelo o actual de avanzar la escena hasta que el usuario haya recolectado la cantidad correcta de frutas. Al concluir la corrección, no es intuitivo si el usuario se equivocó, ni cuando terminó la explicación. En la escena *Verificación*, el uso de la slider para indicar cuántas frutas fueron recolectadas en la escena *Juego* no es intuitiva, por lo cual se sugirió agregar una animación de una

mano arrastrándola, de a modo de instructivo, para ilustrar su uso al usuario. Asimismo, al oprimir botón de ayuda, se deberá agregar audio correspondiente que anuncie las indicaciones de manera verbal, así como un audio que anuncie las frutas a recolectar de manera verbal.

El juego de Vocabulario se reportó funcional, más sin embargo, se hizo notación de agregar una dio de recompensa cuando el usuario concluyera de ordenar una palabra, indicarle al usuario cuando hubiese un cambio de nivel a manera de recompensa, agregar botones de navegación y funcionalidad adicional, como lo es reproducir las instrucciones del minijuego, silenciar la música de fondo, un botón para regresar al menú de juegos, uno para salirse de la aplicación, y uno que le ofrezca una pista al usuario si se encuentra con problemas para resolver la palabra (i.e., una imagen, o un audio que reproduzca la palabra a formar). Adicionalmente, como apoyo a los primeros niveles, se piensa dar la respuesta mediante la reproducción de audio y la lectura de la definición.

5.2.2. Pedagógica

En adición a evaluar la funcionalidad de la aplicación, la evaluación pedagógica examina que su contenido acate con los conocimientos y habilidades del público objetivo. Un cambio mayor en el marco teórico fue el cambio de la canasta en la escena *Juego* por un mantel extendido, en el cual se colocan las frutas recolectadas por el usuario; de esta manera, éstas se mantienen visibles, y proporcionan al usuario la posibilidad de contar las frutas, en lugar de basarse del audio reproducido y de la etiqueta de texto que mantiene el conteo por el usuario. Asimismo, se propuso la

adición de un aviso que señale retroalimentación inmediata cuando el usuario se equivoque (e.g., impedir tomar más frutas de las que le es solicitada, así como impedir que el usuario avance hasta que tenga las frutas correctas recolectadas.

5.2.3. *Focus group*

Tras concluir la demostración del prototipo, la discusión consecutiva con los alumnos de la clase de *Interacción Humano-Computadora* (LIS3032) se concentró en la intuitividad, consistencia, y comodidad de la aplicación. Al preguntarles si la funcionalidad de los botones era reconocible por sus íconos, un estudiante sugirió editar los botones que controlan el flujo del cuadro de diálogo <> por flechas ← → debido a la estandarización de estas últimas en el mundo real, así evitando la posible confusión que podría surgir para nuevos usuarios con el primer diseño sobresimplificado. Otro compañero hizo seguimiento del cuadro de diálogo, cuestionando la eficacia de utilizar un sistema de diálogo en primer lugar, argumentando que, a la edad del público objetivo, uno carece de interés en detenerse a leer cada oración, inclinándose más en imágenes, animaciones, y en general efectos visuales atractivos.

Al consultar la consistencia de los recursos del prototipo y cómo promovían la organización de la misma, la clase resaltó la importancia de rediseñar la interfaz de usuario del juego de Vocabulario tal que coincidieran sus elementos, colores, y fuentes de letra con la sección de Introducción y el juego de Conteo con el objetivo de brindar un sentido de unidad y harmonía a través de la aplicación completa. Asimismo, se hizo la sugerencia de incluir música y actuación de voz más enérgica y

animada durante las escenas de juego, debido a que la música de fondo presente fue percibida como calmada y relajante, y las voces de los personajes carentes de emoción, lo cual contrastaba con el sentimiento que se buscaba inculcar en el usuario en ese momento.

Respecto al desafío que proveen los juegos incluidos en el prototipo y la diversión procedente de este, un estudiante de la clase propuso que se limitaran las palabras que se incluyeran en el juego de Vocabulario, tales que éstas fuesen menor o igual a tres sílabas debido al nivel de lectura estrecho que poseen los usuarios a finales de su educación preescolar. Además, aconsejó reemplazar la definición de la palabra a formar por una imagen de ésta por las mismas razones. Por último, el *focus group* concluyó con la apreciación unánime de la intuitividad y fácil navegación entre las escenas del prototipo, así como el atractivo de sus imágenes y animaciones para apelar al gusto del público objetivo, dejando como única recomendación el utilizar colores más vivos para destacar a los personajes de sus escenas.

5.3. Cambios finales

Con la retroalimentación brindada en las evaluaciones anteriores, se reiteró el diseño y su desarrollo correspondiente para producir una versión mejorada del prototipo, la cual implementa los cambios que se hicieron mención. A continuación, se puede apreciar una vista de la escena *Juego* de la sección de Conteo con una vista responsiva, sustituyendo la canasta al centro de la escena por un mantel de picnic para inculcar al usuario a que cuente las frutas que recolecte. Asimismo, se incluyó un botón con forma de paloma verde para que el usuario pueda indicar que ha terminado de jugar, y continuar a la escena

Verificación en caso de estar correcto, o a la escena *Corrección* en caso contrario (véase Figura 16). Como apoyo durante el juego, si el usuario se equivoca al intentar recolectar más frutas de las que le fueron solicitadas, se desplegará una notificación para indicarle que está recolectando frutas de más (véase Figura 17); seguido, la fruta arrastrada será devuelta automáticamente al árbol de donde provino.

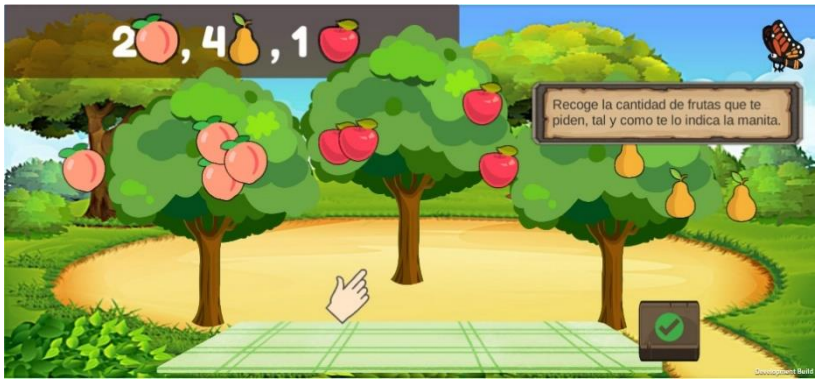


Figura 16. Rediseño de la escena *Juego* de la sección de Conteo, tal que sea responsiva, con la sustitución de la canasta por un mantel de picnic, y un botón en forma de paloma verde con la cual el usuario indica que ha terminado.

Figura 17. Notificación que se despliega en la escena *Juego* cuando el usuario intenta recolectar más frutas de las que le son solicitadas.

De manera similar, la escena *Juego_Vocabulario* fue rediseñada, tal que sea la interfaz de usuario corresponda con el diseño de las demás escenas de la aplicación (i.e., fondo con imagenería del bosque, botones y cuadros de diálogo café), y añadiendo una imagen ilustrativa de la palabra que se busca reordenar (véase Figura 18). Asimismo, se demuestra su funcionalidad para teñir las letras de acuerdo a si fueron arrastradas a las ranuras correctas o no (véase Figura 19).

Figura 18. Rediseño de la escena *Juego_Vocabulario* de la sección de Vocabulario.

Figura 19. Vista de letra acertada (teñida de color verde) y letra equivocada (teñida de color amarillo) durante el juego.

5.4. Evaluación pedagógica pendiente

Habiendo desarrollado una versión mejorada del prototipo de la aplicación, se establecieron planes a futuro para que ésta sea evaluada por expertos en la educación, específicamente aquellos especializados en matemáticas y español para los juegos de Conteo y Vocabulario, respectivamente, así como por los padres, tutores, y usuarios finales a los que va dirigido la aplicación (véase Figura 20).

Figura 20. Grupos de evaluadores que se busca prueben la aplicación en el futuro.

6. Conclusiones

El propósito del proyecto consistió en proveer apoyo a alumnos entre tercero de preescolar y primer grado de primaria con respecto a sus habilidades de lenguaje y matemáticas mediante el refuerzo de su aprendizaje formal a través del uso de una aplicación móvil educativa. Para ello, se plantearon como objetivos el diseño informático de dicha aplicación a partir del marco teórico provisto por el equipo de Pedagogía, el desarrollo subsecuente de un prototipo cuyo contenido abarca las secciones de Introducción, Conteo, y Vocabulario de la aplicación móvil, su evaluación de usabilidad desde el punto de vista técnico y pedagógico, un *focus group* con ocho estudiantes de la materia *Interacción Humano-Computadora* (LIS3032), y finalmente la implementación de mejoras al prototipo en base a la retroalimentación recibida de antedichas evaluaciones.

Conforme a las observaciones realizadas por los *testers* del prototipo, el entregable resultante del proyecto es satisfactorio desde un punto de vista funcional y estético; sin embargo, aún requiere ser pulido en cuanto a la actuación de voz de sus personajes, efectos visuales, y efectos de sonido con el fin de brindar una experiencia más atractiva al público objetivo, y, por ende, aumentar su interés en utilizar la aplicación regularmente y reforzar sus conocimientos a través de este. Asimismo, debido a que el código fuente del prototipo fue escrito a mediados de la licenciatura, no refleja la escalabilidad, estructura, limpieza, y en general calidad de la cual el equipo de desarrollo es capaz de producir actualmente, por lo cual posee áreas de oportunidad en estos y otros ámbitos relacionados al rendimiento de la aplicación.

En cuanto a la herramienta principal de desarrollo, *Unity*, a pesar de ser un motor de videojuegos de excelencia y prestigio en la industria profesional, resultó ser un medio con utilidades excedentes para los requisitos de la aplicación propuesta. Los beneficios principales que ofrece *Unity* por encima del desarrollo nativo (i.e., para una y sólo una plataforma objetivo) son su capacidad de exportar ejecutables de un proyecto a múltiples plataformas, desde sistemas operativos para equipos de cómputo, teléfonos móviles, aplicaciones web, hasta consolas de videojuegos, agilizando el proceso de desarrollo inmensamente; la abstracción de componentes reutilizables estandarizados en el desarrollo de videojuegos (e.g., registro de controles que difieren por plataforma), fenómenos físicos (e.g., cinemática, iluminación, sonido, materiales), y gráficos en 2D y 3D, allanando la curva de aprendizaje, debido a que se requieren conocimientos especializados y de bajo nivel si se desean implementar desde cero; así como una tienda de recursos exclusivos, producto de la comunidad activa y en aumento. Sin embargo, debido a que esta aplicación únicamente busca ser exportada a sistemas operativos *Android*, la gran flexibilidad de plataformas que ofrece *Unity* no se emplea, haciéndolo obsoleto para los objetivos de este proyecto; asimismo, la mayoría de los componentes que ofrece *Unity* tampoco son utilizados en la aplicación, debido a que esta únicamente utiliza pantallas, imágenes, botones efectos de sonido, y funcionalidad *drag and drop*, las cuales son funcionalidades ofrecidas por defecto en desarrollo nativo de *Android*, por lo cual fue posible haber utilizado un motor de videojuegos más ligero, o incluso desarrollar la aplicación de manera nativa para cumplir los objetivos del proyecto.

Este proyecto cumplió como una primera experiencia de trabajo en un equipo multidisciplinario, en la cual el rol de desarrollador requiere entender las necesidades del

público objetivo, aplicar diseño ingenieril de tal manera que pudiesen ser apreciados de manera informática, realizar su implementación, y pasar a la evaluación para deshacerse del sesgo de creador y asegurar que el entregable cumpla con las expectativas y necesidades planteadas. Al seguir el CVDS desde un inicio, el tiempo que se invierte en la planeación y diseño informático de un proyecto se ahorra durante la implementación, debido a que la mayoría de las dudas que hubiesen surgido durante un proyecto enfocado al desarrollo están resueltas de antemano, asegurando una implementación ágil de una aplicación robusta en un menor tiempo y con clientes satisfechos; como se conoce de manera coloquial en el área: "*think twice, code once*."

6.1. Trabajo a futuro

Como trabajo a futuro referente en el ámbito de desarrollo, se sugiere la integración de una base de datos con compatibilidad con el motor de videojuegos *Unity* (e.g., Firebase) para poder registrar usuarios, almacenar su información, y generar estadísticas a partir de sus hábitos de juego para comprender las fortalezas y debilidades de cada usuario, y transmitirles dicho conocimiento a sus padres de familia y educadores para brindarle una atención más personalizada a su formación escolar. Similarmente, a pesar de que *Android* es el sistema operativo móvil mayormente utilizado en el mercado, y por lo tanto fue elegido como la plataforma objetivo del proyecto, la posibilidad que ofrece *Unity* de exportar la aplicación a sistemas operativos complementarios, como iOS y web, abre la posibilidad de ampliar el alcance de la aplicación a más usuarios. Por último, se sugiere refactorizar el código de los controladores, tal que se optimice su comportamiento mediante funciones, y hacer uso de la herencia de clases para englobar atributos y

métodos en común, y con ello evitar clases hinchadas y la redundancia de código. A su pesar, debido a la calidad de código con áreas de mejora y el uso poco eficaz que se está haciendo de *Unity*, se abre la posibilidad de darle un nuevo inicio al proyecto con un desarrollo nativo, o utilizando algún motor de videojuegos ligero o *framework* que permita exportar el proyecto a múltiples plataformas si se desea, en caso de que se llegue a la conclusión de que sea una alternativa más eficaz que refactorizar el código elaborado hasta el momento.

7. Bibliografía

- Cockton, G. (n.d.). *Usability Evaluation*. The Interaction Design Foundation. Recuperado de https://www.interaction-design.org/literature/book/the-encyclopedia-of-human-computer-interaction-2nd-ed/usability-evaluation
- Department of Health and Human Services. (2013, 8 de octubre). *Usability Evaluation Basics*. Usability.gov. Recuperado de https://www.usability.gov/what-and-why/usability-evaluation.html
- Espinosa Plata, M., Maurer Benítez, I., Lozada Vega, S. G. (2019). *Los Números del Bosque* [Trabajo final de curso no publicado]. Universidad de las Américas Puebla.
- hotjar. (2022, 22 de febrero). *The different types of usability testing methods for your projects*. Recuperado de https://www.hotjar.com/usability-testing/methods/
- Kneuper, R. (2017). Sixty Years of Software Development Life Cycle Models. *IEEE Annals of the History of Computing*, Vol. 39 (3), pp. 41 – 54. Recuperado de https://ieeexplore-ieee-org.udlap.idm.oclc.org/stamp/stamp.jsp?arnumber=8047358
- Nuño Mayer, A., Treviño Cantú, J., & Bonilla Rius, E. (2017). Aprendizajes clave para la educación integral. Plan y programas de estudio para la educación básica. Recuperado de https://www.planyprogramasdestudio.sep.gob.mx/descargables/biblioteca/preescolar/1LpM-Preescolar-DIGITAL.pdf

- Unity Technologies. (2022). *Bring your game to life*. Recuperado de https://unity.com/solutions/create-games
- Unity Technologies. (2021, 14 de junio). *IBeginDragHandler.OnBeginDrag*. Recuperado de https://docs.unity3d.com/2018.4/Documentation/ScriptReference/EventSystems.IBeginDragHandler.OnBeginDrag.html
- Unity Technologies. (2021, 14 de junio). *IDragHandler.OnDrag*. Recuperado de https://docs.unity3d.com/2018.4/Documentation/ScriptReference/EventSystems.IDragHandler.OnDrag.html
- Unity Technologies. (2021, 14 de junio). *IDropHandler.OnDrop*. Recuperado de https://docs.unity3d.com/2018.4/Documentation/ScriptReference/EventSystems.IDropHandler.OnDrop.html
- Unity Technologies. (2021, 14 de junio). *IEndDragHandler.OnEndDrag*. Recuperado de https://docs.unity3d.com/2018.4/Documentation/ScriptReference/EventSystems.IEndDragHandler.OnEndDrag.html

8. Anexos

8.1. Anexo 1: Storyboard de *Los Números del Bosque* (base del juego de Conteo)

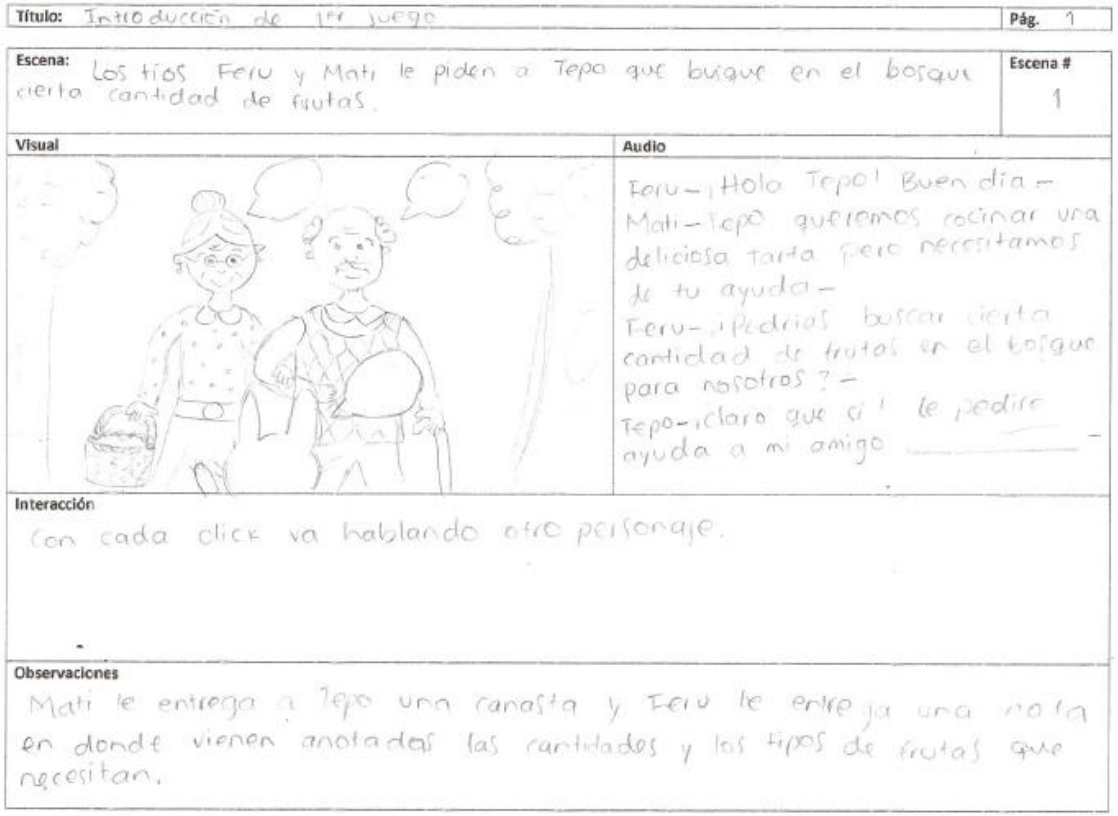


Figura 21. Escena *Introducción* del juego de Conteo.

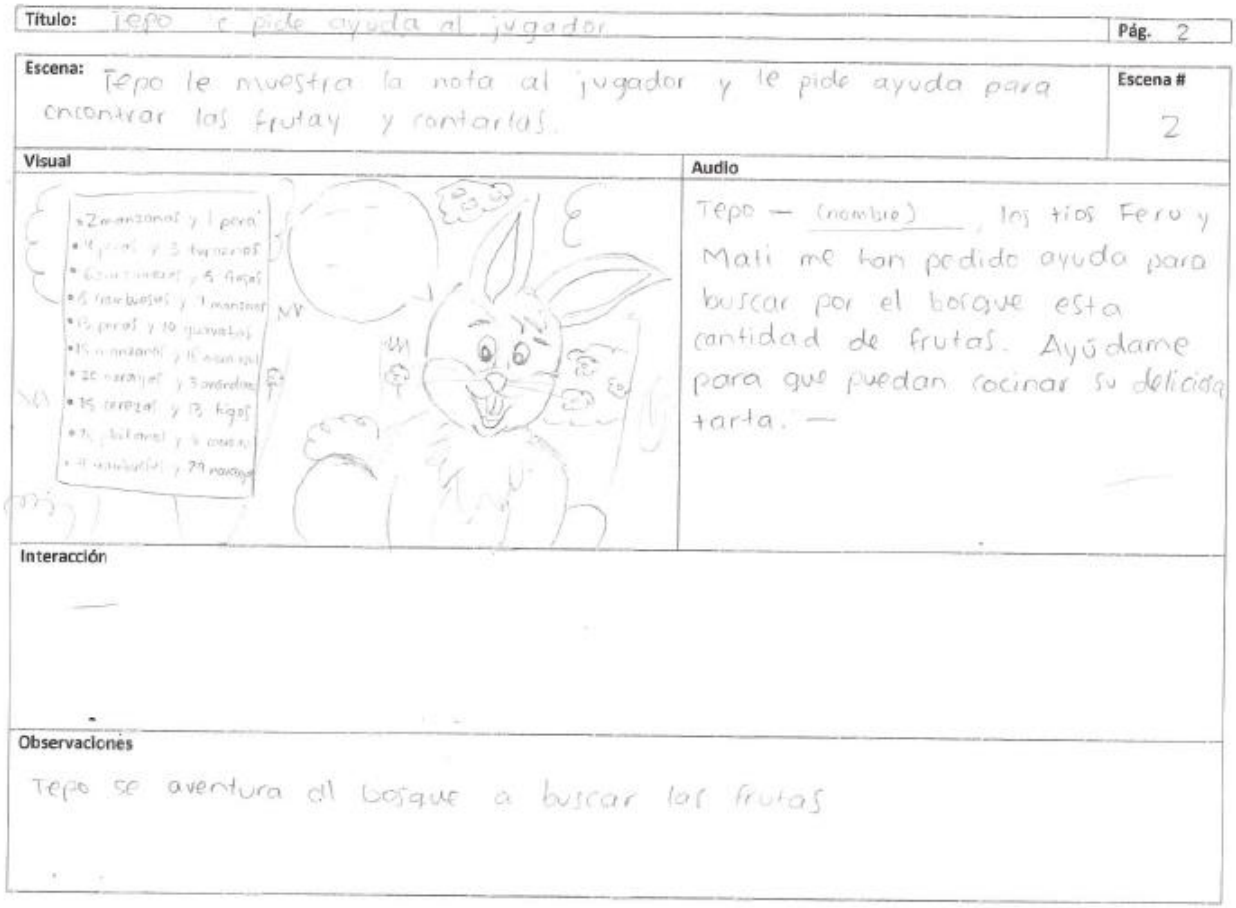


Figura 22. Escena *Selección de Nivel* del juego de Conteo. Originalmente, las cantidades y tipos de frutas eran fijas y no generadas de manera aleatoria.

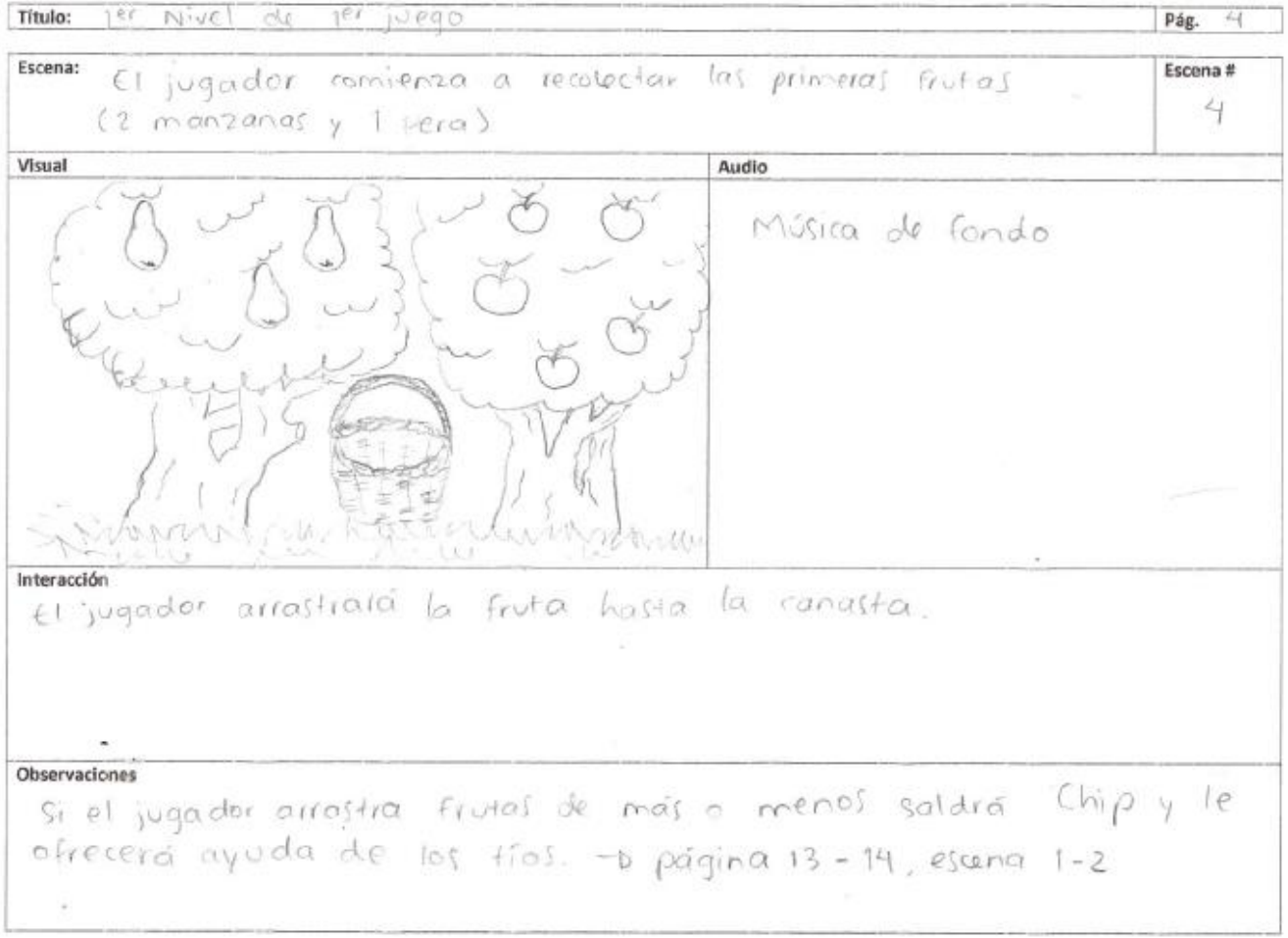


Figura 23. Escena *Juego* del juego de Conteo.

Título: Suma de las frutas (peras y manzanas) Pág. 5

Escena: El jugador deberá de contar las frutas que ha recolectado Escena # 5

Visual	Audio
	Tepo – ¡Genial! ¿Cuántas frutas logramos recolectar en total? ← respuesta (Si la tiene bien) Tepo- Muy bien. Ahora vamos a buscar más frutas. *(Si la tiene mal) Chip- ¿Quieres que los tíos Feru y Mati te ayuden?

Interacción

El jugador pondrá el número de las frutas recolectadas en el espacio blanco.

Observaciones

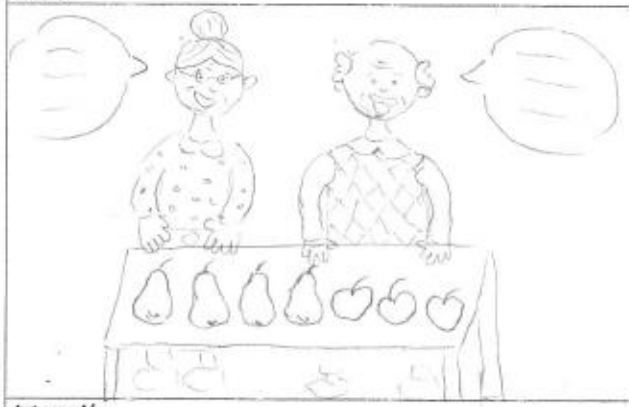
Mientras Tepo habla, las frutas saldrán de la canasta para que el jugador pueda contarlas.
Si lo tiene mal → página 13 - 14, escena 1 - 2

Figura 24. Escena *Verificación* del juego de Conteo.

Título: Ayuda juego 1.2 | Pág. 14

Escena: Los tíos Feru y Mati ayudan al jugador a contar fruta por fruta. | Escena # 2

Visual



Audio

Feru – Estas son 3 manzanas. (las cuenta 1x1)

Mati – Estas son 4 peras. (las cuenta 1x1)

Feru – juntas dan 7 frutas (las cuenta 1x1)

Interacción

—

Observaciones

Cuando se cuente fruta por fruta estas se harán un poco más grandes que las demás y brillarán.

Figura 25. Escena *Corrección* del juego de Conteo.

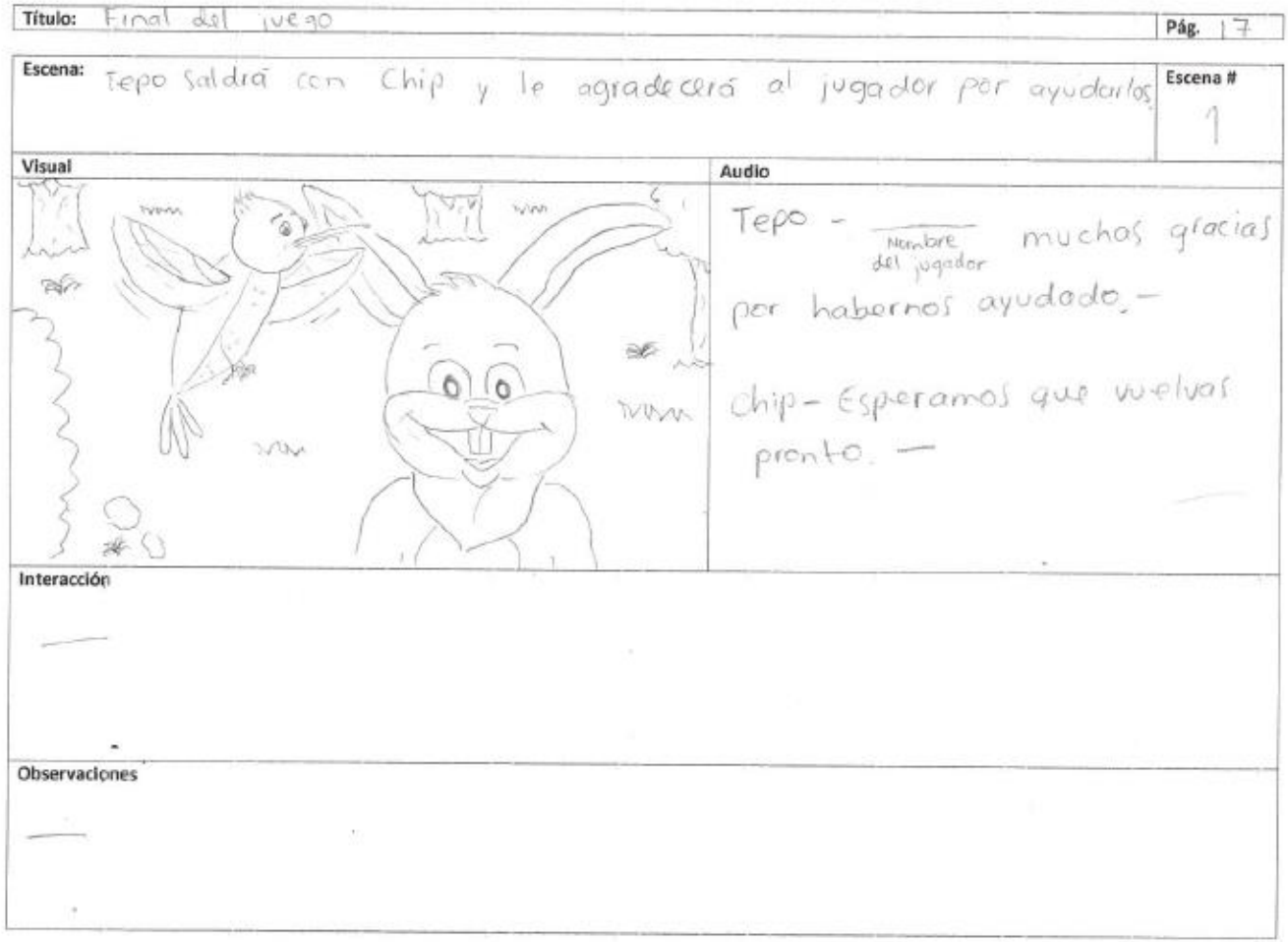


Figura 26. Escena *Agradecimiento* de la sección de Conteo.

8.2. Anexo 2: Evaluación técnica de usabilidad

	>	Cuando avanza escena no detiene audio			>>	
	Tepo: ..		Falta cambiar voz a Tepo diferente del tío			
	(jugador), los tíos...  Ayudame para ...		Audio incorrecto			+ Agregar audio correcto + cambiar (jugador) por el valor de la etiqueta correspondiente
	 Tutorial				Botón para activar las instrucciones	+ Pasarlo al inicio de la app, después de entrar a la sesión con su botón correspondiente para saltar las instrucciones
	 Botón '!'					+ Agregar audio correspondiente cuando se active el botón. Es el que se reproduce en la escena señalada con Audio incorrecto

Tabla 8. Retroalimentación brindada para el minjuego de Conteo.

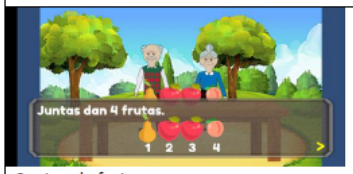
		Revisar esas ‘,’ que las frutas no se salgan ni se escondan			+ Audio con lo que se está pidiendo (recolecta una pera, dos duraznos ..) al inicio del ejercicio + Botón ‘listo’ o ‘palomita’ para que avance cuando se sienta listo aunque no lo haya hecho bien	
	Conteo de la canasta	No funciona, muy ruidoso y confunde el conteo de cada fruta				+Conteo por frutas en la canasta: una manzana, dos manzanas, una pera, tres manzanas
	 Conteo de frutas		Audios desincronizados y empalmados			+ revisar audios grabados, preferible que se escuche una sola voz
			Audio 1 durazno			+revisar audio con género correspondiente, dice ‘una durazno’, el texto: este es

Tabla 9. Retroalimentación brindada para el minjuego de Conteo (continuación).

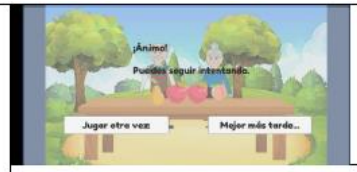
						+ cuando se termina el ejercicio no se sabe si estuvo bien o no y cuando acabó.
		La barra no es intuitiva			+poner la manita que arrastre (a modo de instructivo)	
						

Tabla 10. Retroalimentación brindada para el minjuego de Conteo (continuación).

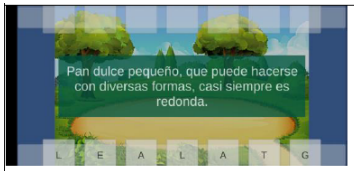
 Vocabulario	funcionan	+Agregar audio de recompensa cuando termina de escribir la palabra			+ Botón con instrucciones + Boton quitar música + Boton salir + Boton regresar + Botón de pista (hint) que aparezca	+Leer la definición y dar la respuesta para primeros niveles (reproducir audio) +mandar las palabras de manera aleatoria
					imagen o diga la palabra +Indicar cambio de nivel (parece que nunca acaba y siempre son las mismas palabras)	

Tabla 11. Retroalimentación brindada para el minjuego de Vocabulario.

8.3. Anexo 3: Evaluación pedagógica

Juego de Conteo			
Mati: Tepo, queremos cocinar...		Agregar flecha para regresar en texto.	
Feru: ¿Podrías buscar cierta cantidad...		Agregar flecha para regresar en texto.	
Tepo: ¡Claro que sí!		¡Claro que sí! Pediré ayuda.	
(Jugador), los tíos Feru		Quitar (Jugador), corregir el llamado a archivo de audio correcto. El nombre	

Tabla 12. Retroalimentación brindada para el minjuego de Conteo.

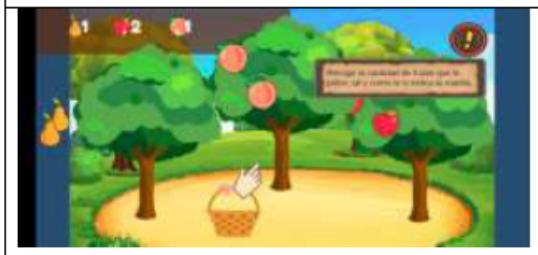
Pantalla	Aciertos	Cambios	Sugerencias
		del archivo es: Tepo2_LosTiod.aac	
		Es mejor usar un mantel extendido sobre el que se coloquen las frutas en lugar de la canasta, ya que así están siempre visibles para el usuario. Eso le ayuda en una etapa de desarrollo en la que todavía no desarrolla pensamiento abstracto.	En lugar de usar el botón con una exclamación, sería mejor un signo de interrogación que es el que normalmente se utiliza para ayuda. Sería mejor que el texto de ayuda solo aparezca cuando se seleccione el botón de ayuda. El mantel estaría extendido en el suelo donde está el claro del bosque. 
			Cuando se equivoca no es claro, debe haber algo que le señale que dé retroalimentación inmediata. Por ejemplo si toma más de la cantidad requerida de una fruta, debe haber una señal auditiva como "¡oh, oh!" y esa fruta adicional debe regresar al árbol. El usuario

Tabla 13. Retroalimentación brindada para el minjuego de Conteo (continuación).

Pantalla	Aciertos	Cambios	Sugerencias
			no debe poder avanzar hasta que lo tenga correcto.
Slider del total de frutas. Retroalimentación de los tíos Feru y Mati.	Que los tíos Feru y Mati den retroalimentación cuando se equivoca en el total de frutas del slider está muy bien.		Igual en el slider, si se equivoca, que aparezca la señal auditiva de error.
Vocabulario		El menú de "hamburguesita" es fundamental para poder parar o salir. Ajustar menú de juegos para que este sea el último, ya que tiene un nivel de dificultad mucho mayor.	Dado que para ordenar letras, implica que ya sabe leer las palabras completas, sería mejor hacer grupos por tamaños de palabras para que de ahí se elija al azar entre las más pequeñas primero, y una vez completado el grupo se pase a otro grupo de palabras con más letras, hasta llegar a las más grandes.

Tabla 14. Retroalimentación brindada para el minjuego de Conteo (continuación) y para el minjuego de Vocabulario.