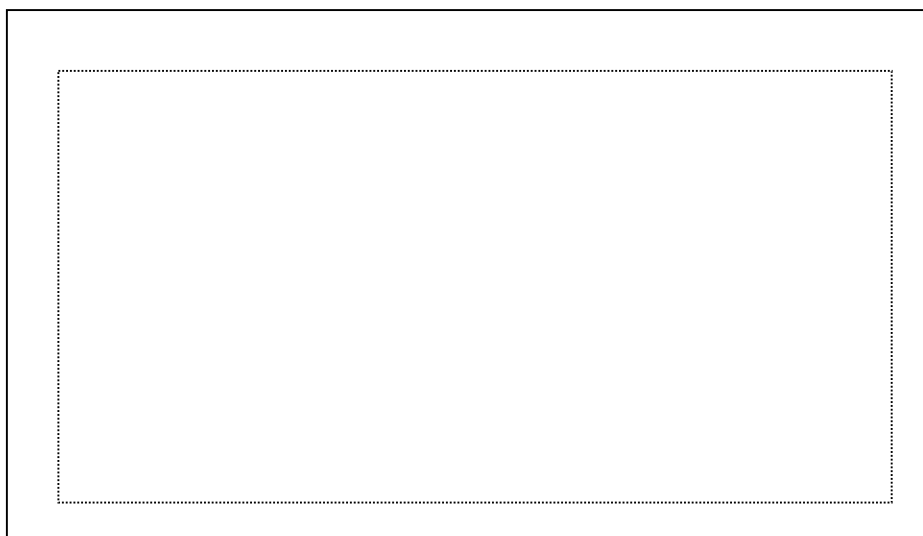




UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE CIENCIAS
ESCUELA DE COMPUTACIÓN



CARACAS - VENEZUELA



Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación
Computación Gráfica

Aplicación Android para iniciar a los niños de 3 a 6 años en el proceso de lectura.

Trabajo Especial de Grado presentado ante la Ilustre
Universidad Central de Venezuela por el Bachiller
Carlos Miguel Carvajal Lepplaid
para optar al título de Licenciado en Computación

Tutor: Prof. Rhadamés Carmona

Caracas, 10 de Agosto de 2015

Caracas, Agosto de 2015
Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación



ACTA DEL VEREDICTO

Quienes suscriben, Miembros del Jurado designado por el Consejo de la Escuela de Computación para examinar el Trabajo Especial de Grado, presentado por el Bachiller Carlos Miguel Carvajal Lepplaid C.I.: V-14.225.551, con el título "Aplicación Android para iniciar a los niños de 3 a 6 años en el proceso de lectura", a los fines de cumplir con el requisito legal para optar al título de Licenciado en Computación, dejan constancia de lo siguiente:

Leído el trabajo por cada uno de los Miembros del Jurado, se fijó el día 10 de Agosto de 2015, a las 10am, para que su autor lo defendiera en forma pública, en el Centro de Computación Gráfica, lo cual este realizó mediante una exposición oral de su contenido, y luego respondió satisfactoriamente a las preguntas que les fueron formuladas por el Jurado, todo ello conforme a lo dispuesto en la Ley de Universidades y demás normativas vigentes de la Universidad Central de Venezuela. Finalizada la defensa pública del Trabajo Especial de Grado, el jurado decidió aprobarlo.

En fe de lo cual se levanta la presente acta, en Caracas el lunes diez (10) de agosto de 2015 dejándose también constancia de que actuó como Coordinador del Jurado el Profesor Tutor Rhadamés Carmona.

A handwritten signature in dark ink, appearing to be 'Rhadamés Carmona', written over a horizontal line.

Prof. Rhadamés Carmona
(Tutor)

A large, stylized handwritten signature in dark ink, appearing to be 'Yusneyi Carballo', written over a horizontal line.

Prof. Yusneyi Carballo
(Jurado Principal)

A handwritten signature in dark ink, appearing to be 'Francisco Moreno', written over a horizontal line.

Prof. Francisco Moreno
(Jurado Principal)

DEDICATORIA

Le dedico este trabajo en primer lugar a mis hijos Carlos Miguel y José Miguel, como demostración de que si trabajamos dedicadamente en una causa buena, podemos alcanzar nuestras metas.

Dedico este trabajo a mi esposa Belkis María, quien me ha acompañado a lo largo de toda esta carrera, animándome y entregándome todo su amor y apoyo.

También dedico este trabajo a mis padres quienes también ayudaron en el logro de esta meta.

Carlos Miguel Carvajal

AGRADECIMIENTOS

En primer lugar deseo agradecer a mi Padre Celestial por todas las bendiciones que me ha dado, entre ellas el poder graduarme en esta carrera.

Gracias a mi esposa Belkis María Páez quien me ha apoyado constantemente a lo largo de gran parte de mi vida.

Gracias a mis hijos Carlos Miguel y José Miguel por darme el ánimo, la alegría y la motivación para ofrecerles un mejor futuro.

Gracias a mis padres Ilse Lepplaid y Leonardo Carvajal porque a pesar de ser yo un hombre no tan joven, siempre me apoyaron con muchas de las cosas necesarias a lo largo de la carrera.

Gracias a mi tutor Prof. Rhadamés Carmona por la paciencia y la confianza puesta en mí y en este trabajo.

Gracias a todos mis familiares, amigos, profesores, y a todos aquellos que de alguna u otra manera estuvieron involucrados a lo largo de esta carrera y que por razones de espacio no puedo agregar sus nombres aquí; pero que recuerdo y aprecio constantemente. Su servicio, ánimo e inspiración contribuyeron en gran medida a que esta meta se haya logrado.

¡Gracias a todos ustedes!

Carlos Miguel Carvajal.



Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación
Centro de Computación Gráfica

Aplicación Android para iniciar a los niños de 3 a 6 años en el proceso de lectura.

Autor: Carlos Miguel Carvajal Lepplaid

C.I. 14.225.551

carloscarvajal0007@yahoo.com

Tutor: Prof. Rhadamés Carmona

rhadames.carmona@ciens.ucv.ve

Fecha: 10/08/2015

RESUMEN

Las tecnologías de dispositivos móviles (*Smartphones* y *Tablets*), que en un principio se comercializaron principalmente para facilitar la comunicación y el entretenimiento de las personas, han llegado a desempeñar un papel importante como apoyo a los procesos de enseñanza y aprendizaje en cualquier momento y lugar. Y es que dichos dispositivos les resultan sumamente atractivos a los niños ya que de forma natural e intuitiva estimulan la exploración, lo cual es una actitud muy característica de ellos. De estas consideraciones surge la idea de desarrollar una aplicación móvil diseñada como herramienta de apoyo a la educación formal para la iniciación en el proceso de lectura, dirigida a niños de edades comprendidas entre tres y los seis años. Se utilizó el Método Mixto de aprendizaje de la lectura, en el cual se relacionaron imágenes con los sonidos de palabras de uso frecuente en el español.

Palabras clave: *Android*, enseñanza, lectura, aplicación, móvil, *Smartphone*, *Tablet*, Tableta, *Unity*.

ÍNDICE GENERAL

DEDICATORIA	i
AGRADECIMIENTOS	ii
RESUMEN	iii
ÍNDICE DE FIGURAS	v
ÍNDICE DE TABLAS	x
INTRODUCCIÓN	1
MARCO TEÓRICO	3
1.1 Teorías del aprendizaje cognitivo	4
1.1.1 El Conductismo	4
1.1.2 El Constructivismo	5
1.1.3 La Programación Neurolingüística	8
1.1.3.1. Los Sistemas de Representación	9
1.1.3.2 Las Señales de Acceso Ocular	10
1.1.4 La Teoría Uno	13
1.1.5 Teoría Cognitiva del Aprendizaje Multimedia	14
1.2 Características evolutivas de los niños según su edad	15
1.2.1 El niño de tres años	15
1.2.2 El niño de cuatro años	16
1.2.3 El niño de cinco años	17
1.2.4 El niño de seis años	18
1.3 Métodos de enseñanza de la lectura	18
1.3.1 Método Sintético o Silábico	19
1.3.2 Método Analítico o Global	19
1.3.3 Métodos Mixtos	20
1.4 Teorías sobre el diseño de software educativo	21
1.4.1 Metodología para el Desarrollo de Software Educativo (DESED)	21
1.4.2 El Modelo Sistemático.	25
1.4.3 Modelos no lineales	26
CAPÍTULO II:	28

TRABAJOS PREVIOS	28
2.1 Trabajo: “Análisis de las Características y uso de software educativo para niños en edad Inicial”	28
2.2 Trabajo: “Süchiki Walekerü: un ejemplo del uso de las TIC en escuelas indígenas. Caso Wayuu”	29
2.3 Trabajo: “El aprendizaje de las matemáticas en segundo grado de primaria por medio de dispositivos móviles”	32
2.4 Trabajo: “Juegos educativos en dispositivos móviles”	33
2.5 Trabajo: “Los Efectos de la Tecnología Multimedia sobre el Aprendizaje”	36
2.6 Trabajo: “Aplicaciones móviles: Juegos que transforman la educación”	37
CAPÍTULO III:	41
PROPUESTA DE TRABAJO ESPECIAL DE GRADO	41
3.1 Planteamiento del problema	41
3.2 Justificación	42
3.3 Objetivo general	43
3.4 Objetivos específicos	43
3.5 Alcance	44
3.6 Análisis del proyecto	44
3.7 Requisitos funcionales (estrategia de la solución):	45
3.7.1 Clases y actividades	45
3.7.2 Mini-juegos	48
3.7.3 Menú principal	49
3.8. Requisitos no funcionales	50
3.8.1 Documentación	50
3.8.2 Seguridad	50
3.8.3 Actualización y portabilidad	50
3.8.4 Interfaz y usabilidad	50
3.8.5 Rendimiento	51
3.9 Metodología de desarrollo	51
3.10 Tecnologías a utilizar	52
3.10.1 Lenguajes de programación	52
3.10.2 Almacenamiento de datos	52
3.10.3 Interfaz	53

3.10.4 Plataforma de desarrollo y pruebas	54
3.11 Planificación inicial del proyecto	55
3.11.1 Casos de uso.....	61
3.11.2 Plan de entrega	67
3.12 Primera iteración	67
3.12.1 Planificación.....	67
3.12.1.1 Historias de usuario	67
3.12.1.2 Plan de entrega.....	69
3.12.1.3 Velocidad del proyecto.....	69
3.12.1.4 Tareas específicas.....	69
3.12.1.5 Reuniones.....	73
3.12.2 Diseño	73
3.12.2.1 Diagrama de clases (<i>GameObjects</i>)	75
3.12.3 Codificación	77
3.12.4 Pruebas	77
3.13 Segunda iteración	80
3.13.1 Planificación.....	80
3.13.1.1 Historias de usuario	80
3.13.1.2. Plan de entrega.....	80
3.13.1.3. Velocidad del proyecto.....	81
3.13.1.4. Tareas específicas.....	81
3.13.1.5 Reuniones.....	82
3.13.2 Diseño	82
3.13.2.1. Diagrama de clases	82
3.13.3 Codificación	82
3.13.4 Pruebas	83
3.14 Tercera iteración	84
3.14.1 Planificación.....	84
3.14.1.1 Historias de usuario	85
3.14.1.2. Plan de entrega.....	86
3.14.1.3. Velocidad del proyecto.....	86
3.14.1.4. Tareas específicas.....	86
3.14.1.5 Reuniones.....	89

3.14.2 Diseño	89
3.14.2.1 <i>Fruit Ninja</i>	90
3.14.2.2 <i>Roll a Ball</i>	94
3.14.2.3 <i>Circuit Racer</i>	98
3.14.2.3 <i>Angry Letters</i>	103
3.15 Cuarta iteración	108
3.15.1 Planificación.....	108
3.15.1.1 Historias de usuario	108
3.15.1.2 Plan de entrega.....	110
3.15.1.3 Velocidad del proyecto.....	110
3.15.1.4 Tareas específicas.....	110
3.15.1.5 Reuniones.....	113
3.15.2 Diseño	113
3.15.2.1 Diagramas de Clases (<i>GameObjects</i>).....	113
3.15.2.3 Codificación.....	118
3.15.2.4 Pruebas	119
3.16 Quinta iteración	121
3.16.1 Planificación.....	122
3.16.1.1 Historias de usuario	122
3.16.1.2 Plan de entrega.....	123
3.16.1.3 Velocidad del proyecto.....	123
3.16.1.4 Tareas específicas.....	123
3.16.1.5 Reuniones.....	125
3.16.2 Diseño	126
3.16.3 Codificación	126
3.16. 4 Pruebas	126
CAPITULO IV:	127
EVALUACIÓN	127
Resultados	129
CAPÍTULO V:	131
CONCLUSIONES, RECOMENDACIONES Y TRABAJOS FUTUROS.....	131
APÉNDICE A: DIAGRAMAS DE CLASES (<i>GAMEOBJECTS</i>).....	134
- A.1 Diagramas de clases para la aplicación en general:	134

- A.2 Diagramas de clases del Mini-juego <i>Fruit Ninja</i>	141
- A.3 Diagramas de clases del Mini-juego <i>Roll a Ball</i>	143
- A.4 Diagramas de clases del Mini-juego <i>Circuit Racer</i>	144
- A.5 Diagramas de clases del Mini-juego <i>Angry Letters</i>	146
- A.6 Diagramas de clases de pertenecientes a la cuarta iteración.....	150
APÉNDICE B: MÉTODOS PARA EL DESARROLLO DE SOFTWARE	154
B.1 Métodos Ágiles	155
B.1.1 <i>Adaptive Software Development</i> (Desarrollo Adaptativo de Software, ASD)	156
B.1.2 <i>Agile Unified Process</i> (Proceso Ágil Unificado, AUP)	157
B.1.3 <i>Feature-driven development</i> (Desarrollo Orientado a las Características, <i>FDD</i>).....	157
B.1.4 <i>Extreme Programming</i> (Programación Extrema, XP).....	157
B.1.5 <i>Scrum</i>	158
B.2 Características de Extreme Programming (XP).....	159
B.3 Actores en Extreme Programming (XP)	159
B.4 Fases del Extreme Programming (XP).....	160
B.5 Planificación del Proyecto	161
B.6 Diseño de la Aplicación.....	161
B.7 Implementación de la Aplicación.....	162
B.8 Pruebas a la Aplicación.....	163
B.9 Buenas Prácticas en XP	163
APÉNDICE C: <i>ANDROID DEVELOPER TOOLS</i> Y <i>UNITY</i>	166
C.1 Android Developer Tools (Herramientas para Desarrolladores en Android, ADT):	166
C.2 Unity	168
C.2.1 Características de <i>Unity</i>	170
APÉNDICE D: MARCO LEGAL	176
D.1 CONSTITUCIÓN DE LA REPÚBLICA BOLIVARIANA DE VENEZUELA....	176
D.2 LEY ORGÁNICA DE EDUCACIÓN.....	177
D.3 LEY ORGÁNICA DE CIENCIA, TECNOLOGIA E INNOVACIÓN.....	179
REFERENCIAS	181

ÍNDICE DE FIGURAS

	Pág.
CAPITULO I	
<u>Figura I-1. Sistemas de Representación</u>	10
<u>Figura I-2. Componente Visual</u>	11
<u>Figura I-3. Componente Auditivo</u>	12
<u>Figura I-4. Componente Kinestésico</u>	12
<u>Figura I-5. Esquema general de la Metodología DESED</u>	25
<u>Figura I-6. Modelo Sistemático para la selección o desarrollo de MEC desarrollado por Galvis</u>	26
<u>Figura I-7. Metodología Thales</u>	27
CAPITULO II	
<u>Figura II-1. Impresión de pantalla 1 del software Söchiki Walekerü</u>	30
<u>Figura II-2. Impresión de pantalla 2 del software Söchiki Walekerü</u>	31
<u>Figura II-3. Impresión de pantalla 1 del software de Aquino A. Zahedi</u>	32
<u>Figura II-4. Impresión de pantalla 2 del software de Aquino A. Zahedi</u>	33
<u>Figura II-5. Impresión de pantalla 1 del software de Villoria, Martín y de las Cuevas</u>	34
<u>Figura II-6. Impresión de pantalla 2 del software de Villoria, Martín y de las Cuevas</u>	35
<u>Figura II-7. Impresión de pantalla 1 del software de Zhang E. y Loeb L</u>	38
<u>Figura II-8. Impresión de pantalla 2 del software de Zhang E. y Loeb L</u>	39
<u>Figura II-9. Impresión de pantalla 3 del software de Zhang E. y Loeb L</u>	40
CAPITULO III	
<u>Figura III-1. Representación gráfica de la elaboración de prototipos</u>	52
<u>Figura III-2. Caso de uso: Menú Principal</u>	61
<u>Figura III-3. Caso de uso: Aprende ya/ Continuar</u>	63
<u>Figura III-4. Caso de uso: Seleccionar clase</u>	63
<u>Figura III-5. Impresión de pantalla de la organización del proyecto, en el editor de Unity</u>	74

<u>Figura III-6. Diagrama del sistema</u>	74
<u>Figura III-7. Diagrama de GameObjects para las actividades</u>	76
<u>Figura III-8. Impresión de pantalla de prueba de la actividad 1 “toca cada vocal/sílaba 2 veces”</u>	78
<u>Figura III-9. Impresión de pantalla de prueba de la actividad 2 “toca la vocal/sílaba”</u>	78
<u>Figura III-10. Impresión de pantalla de prueba de la actividad 3 “toca cada palabra 2 veces”</u>	78
<u>Figura III-11. Impresión de pantalla de la prueba de la actividad 4 “toca cada oración 3 veces”</u>	79
<u>Figura III-12. Impresión de pantalla de la prueba de la actividad 1 “toca cada vocal/sílaba 2 veces” (para el caso de la clase “c”)</u>	79
<u>Figura III-13. Impresión de pantalla de prueba de la actividad 2 “toca la vocal/sílaba” (para el caso de la clase “c”)</u>	79
<u>Figura III-14. Impresión de pantalla de la carga de variables multimedia, en el editor de Unity</u>	83
<u>Figura III-15. Impresión de pantalla de una actividad con todos sus elementos cargados, en el simulador de Unity</u>	84
<u>Figura III-16. Impresión de pantalla de otra actividad con todos sus elementos cargados, en el simulador de Unity</u>	84
<u>Figura III-17. Diagrama de GameObjects del mini-juego Fruit Ninja</u>	91
<u>Figura III-18. Impresión de pantalla del código del mini-juego Fruit Ninja, en MonoDevelop de Unity</u>	92
<u>Figura III-19. Impresión de pantalla del mini-juego Fruit Ninja, en el editor de Unity</u>	93
<u>Figura III-20. Impresión de pantalla del mini-juego Fruit Ninja, en el simulador de Unity</u>	93
<u>Figura III-21. Diagrama de GameObjects del mini-juego Roll a Ball</u>	95
<u>Figura III-22. Impresión de pantalla del código PlayerController1.cs*del mini-juego Roll a Ball, en MonoDevelop de Unity</u>	96
<u>Figura III-23. Impresión de pantalla del mini-juego Roll a Ball, en el editor de Unity</u>	97

<u>Figura III-24. Impresión de pantalla del mini-juego Roll a Ball, en el simulador de Unity.....</u>	98
<u>Figura III-25. Diagrama de GameObjects Main Menu del mini-juego Circuit Racer.....</u>	99
<u>Figura III-26. Diagrama de GameObjects Pick Car Menu del mini-juego Circuit Racer.....</u>	99
<u>Figura III-27. Diagrama de GameObjects Pick Diff Menu del mini-juego Circuit Racer.....</u>	100
<u>Figura III-28. Diagrama de GameObjects del juego principal en el mini-juego Circuit Racer.....</u>	100
<u>Figura III-29. Impresión de pantalla del código CarController.cs*del mini-juego Circuit Racer, en MonoDevelop de Unity.....</u>	101
<u>Figura III-30. Impresión de pantalla del mini-juego Circuit Racer, en el editor de Unity.....</u>	102
<u>Figura III-31. Impresión de pantalla del mini-juego Circuit Racer, en el simulador de Unity.....</u>	103
<u>Figura III-32. Diagrama de GameObjects Main Menu del mini-juego Angry Letters.....</u>	105
<u>Figura III-33. Impresión de pantalla del mini-juego Angry Letters, en el editor de Unity.....</u>	107
<u>Figura III-34. Impresión de pantalla del mini-juego Angry Letters, en el simulador de Unity.....</u>	107
<u>Figura III-35. Diagrama de GameObjects del Menu Principal.....</u>	114
<u>Figura III-36. Diagrama de GameObjects del Menu “Seleccionar clase”.....</u>	115
<u>Figura III-37. Diagrama de GameObjects del menú “Reiniciar app”.....</u>	116
<u>Figura III-38. Diagrama de GameObjects del menú “Créditos”.....</u>	117
<u>Figura III-39. Diagrama de GameObjects del menú “Disclaimer”.....</u>	118
<u>Figura III-40. Impresión de pantalla de prueba de la pantalla “Disclaimer”.....</u>	119
<u>Figura III-41. Impresión de pantalla del menú principal de la aplicación.....</u>	119
<u>Figura III-42. Diagrama de GameObjects del menú “Clases”.....</u>	120
<u>Figura III-43. Diagrama de GameObjects del menú “Reiniciar App”.....</u>	120
<u>Figura III-44. Diagrama de GameObjects de la función “Reiniciar App”.....</u>	121

Figura III-45. Diagrama de GameObjects del menú “Créditos”	121
--	-----

CAPITULO IV

Figura IV-1. Plantilla de evaluación N° 1	128
---	-----

Figura IV-2. Plantilla de evaluación N° 2	129
---	-----

Figura IV-3. Plantilla de evaluación N° 3	129
---	-----

APÉNDICE A

Figura A-1. Clase “Cinco Silabas”	135
---	-----

Figura A-2. Clase “TocalaCinco”	137
---	-----

Figura A-3. Clase “TresSilabas”	138
---	-----

Figura A-4. Clase “TocaLaTres”	139
--	-----

Figura A-5. Clase “TresPalabras”	140
--	-----

Figura A-6. Clase “UnaOracion”	141
--	-----

Figura A-7. Clase “MusicLoop”	141
---	-----

Figura A-8. Clase “Fruit Spawner”	142
---	-----

Figura A-9. Clase “Timer”	142
---	-----

Figura A-10. Clase “Lines Handler”	142
--	-----

Figura A-11. Clase “Splash”	142
---	-----

Figura A-12. Clase “Apple”	143
--	-----

Figura A-13. Clase “Player Controller 1”	144
--	-----

Figura A-14. Clase “Camera Follow”	144
--	-----

Figura A-15. Clase “Rotator”	144
--	-----

Figura A-16. Clase “Game Controller”	145
--	-----

Figura A-17. Clase “Car Controller”	145
---	-----

Figura A-18. Clase “Analog Controller”	146
--	-----

Figura A-19. Clase “BGSound”	146
--	-----

Figura A-20. Clase “GUI Controller”	146
---	-----

Figura A-21. Clase “RetainedUserPicksScript”	146
--	-----

Figura A-22. Clase “Game Manager”	147
---	-----

Figura A-23. Clase “Slingshot”	148
--	-----

Figura A-24. Clase “Enemy”	148
--	-----

Figura A-25. Clase “Wood Ball”	148
--	-----

Figura A-26. Clase “Bloque”	148
---	-----

<u>Figura A-27. Clase “Camera Controller”</u>	149
<u>Figura A-28. Clase “Camera Fix Aspect Ratio”</u>	149
<u>Figura A-29. Clase “Camera Follow Angry”</u>	149
<u>Figura A-30. Clase “Camera Move”</u>	149
<u>Figura A-31. Clase “Camera Pinch To Zoom”</u>	149
<u>Figura A-32. Clase “Constants”</u>	150
<u>Figura A-33. Clase “Destructor”</u>	150
<u>Figura A-34. Clase “Enums”</u>	150
<u>Figura A-35. Clase “Menu Script”</u>	150
<u>Figura A-36. Clase “Main Menu Buttons 2”</u>	151
<u>Figura A-37. Clase “Main Menu Music”</u>	151
<u>Figura A-38. Clase “Save Game Script”</u>	151
<u>Figura A-39. Clase “Menu Clases”</u>	151
<u>Figura A-40. Clase “Cargar Escena”</u>	152
<u>Figura A-41. Clase “Error Clases”</u>	152
<u>Figura A-42. Clase “App Reset”</u>	152
<u>Figura A-43. Clase “App Reset 2”</u>	153
<u>Figura A-44. Clase “Fade IN”</u>	153
<u>Figura A-45. Clase “Fade OUT”</u>	153
APÉNDICE B	
<u>Figura B.1. Fases de Desarrollo de la Programación Extrema</u>	160
<u>Figura B.2. Estructura de una Tarjeta de Historia de Usuario</u>	162

ÍNDICE DE TABLAS	Pág.
Capítulo III	
<u>Tabla N°1. Estructura de las clases</u>	47
<u>Tabla N°2. Distribución de los mini-juegos en la aplicación</u>	49
Capítulo IV	
<u>Tabla N° 3. Descripción del equipo de pruebas de la aplicación</u>	127

INTRODUCCIÓN

Vivimos en una era tecnológica. Los cambios que en materia tecnológica se han producido y que se producirán influyen directamente en casi todos los ámbitos de nuestras vidas, como la forma en la que nos relacionamos unos con otros, el uso de nuestro tiempo libre, nuestra forma de aprender y enseñar y, en definitiva, en la forma en la que nos desenvolvemos en el mundo en general.

Los niños están en constante contacto con la tecnología y muchos de ellos antes de aprender a hablar ya juegan en los dispositivos móviles de sus progenitores. Pensando en el tiempo libre y la facilidad casi innata e intuitiva que poseen los niños de hoy día para manejar dichos dispositivos, se desarrolló una aplicación móvil para ayudar en el proceso de enseñanza de la lectura, a niños de edades comprendidas entre los tres y los seis años. Se escogió el tema de la enseñanza de la lectura ya que, esta favorece la adquisición del lenguaje verbal, nuestra forma más importante de comunicación.

En esta era tecnológica, aprender a leer y escribir son elementos esenciales de supervivencia. El carácter móvil de la aplicación ofrecerá la ventaja de poder usarla en cualquier lugar y a cualquier hora que se desee.

La interacción del niño con las imágenes y sonidos fortalece la memoria visual y auditiva, el acatamiento de indicaciones, así como también favorecen los procesos de atención y la comprensión del entorno y desarrollan la posibilidad de elección, siendo así una herramienta didáctica de apoyo para los padres y docentes, y de refuerzo cognitivo del niño, estimulando el aprendizaje autónomo, interesándolo e invitándolo a experimentar. Adicionalmente mejorará la coordinación motriz, psicomotriz y en general sus habilidades en el uso de los dispositivos móviles.

Este trabajo está dividido en tres capítulos e incluye tres apéndices, tal como se indica a continuación:

- En el Capítulo I se explica el marco teórico de este trabajo, el cual incluye la investigación, citas y análisis de múltiples trabajos y desarrollos realizados en el campo de la Educación.
- En el Capítulo II se presentan algunos trabajos previos de importancia, relacionados con este trabajo especial de grado.
- En el Capítulo III se describe la propuesta de Trabajo Especial de Grado como tal, la metodología de desarrollo, descripción de características y listado de requisitos requeridos para tal fin.
- En el Capítulo IV se describe las pruebas y los resultados de la evaluación del prototipo de la aplicación.
- En el Capítulo V se detallan las conclusiones y recomendaciones de esta investigación, así como también algunas sugerencias para posibles trabajos futuros en esta línea de investigación.
- En el Apéndice A se presentan las clases detalladas de los diagramas de Clases (*GameObjects*) del capítulo III.
- En el Apéndice B se describen los principales métodos de desarrollo de software, así como las características más resaltantes del método elegido para este trabajo (*Extreme Programming - XP*).
- En el Apéndice C se explican las dos principales plataformas de desarrollo a ser utilizadas para el desarrollo del software descritas en este trabajo, las cuales son *Android Developer Tools (ADT)* y *Unity*.
- En el Apéndice D se presenta el marco legal que da soporte el desarrollo educativo y el manejo de las tecnologías relacionadas, en Venezuela.

CAPÍTULO I:

MARCO TEÓRICO

Esta investigación está dirigida a niños de edades comprendidas entre los tres y seis años. Por eso, es necesario hacer una revisión bibliográfica de algunas Teorías del Aprendizaje Cognitivo como el Conductismo, el Constructivismo, La Programación Neurolingüística (PNL), la Teoría Uno y Teoría Cognitiva del Aprendizaje Multimedia. Seguidamente se estudian las características evolutivas de los niños, sus habilidades, intereses y nivel de desarrollo según su edad. Todos estos aspectos son importantes y necesarios tenerlos en cuenta en el momento de diseñar una aplicación móvil.

Más adelante se describen los Métodos de Enseñanza de la lectura, la cual es una parte fundamental en el desarrollo intelectual de cada ser humano. La lectura aumenta el vocabulario de los niños así como también su capacidad de opinar, comparar y sacar conclusiones. Adicionalmente, en esta era informática, aprender a leer y escribir son herramientas fundamentales de supervivencia. Según Borrero L.

(5).

Leer es esencial para ser exitoso y competitivo en el mundo de hoy. La capacidad de leer y escribir es una herramienta poderosa del pensamiento, es la puerta virtual a otros tipos de aprendizaje, brinda la posibilidad de buscar información, expresar la opinión y tomar posiciones. El poder leer abre toda una perspectiva de posibilidades de la vida, desde el placer del entretenimiento, hasta la investigación y el trabajo. El analfabeto está en franca desventaja hoy más que nunca (p.34)

Posteriormente se abordan las Teorías sobre el Diseño de Softwares Educativos, sus bondades, sus fases y finalmente la escogencia de una teoría que mejor se adapte a este proyecto.

1.1 Teorías del aprendizaje cognitivo

El crear un software de tipo educativo se inicia con la elección de un modelo de Aprendizaje. Esta elección se hace de acuerdo al tipo de aprendizaje que se desee producir en los usuarios finales del mismo. Dichas teorías le ofrecen al diseñador estrategias y técnicas validadas para facilitar aprendizajes. Existen numerosas teorías del Aprendizaje; en este trabajo sólo se mencionan algunas de ellas.

1.1.1 El Conductismo

Existen varios representantes de esta corriente cognitiva. Watson (1878-1958) es uno de ellos, quien planteó que el desarrollo se produce a partir de pequeños incrementos. Se puede incentivar el aprendizaje de un comportamiento dado un conjunto de estímulos, a través de un refuerzo positivo, siendo este la clave para el aprendizaje. En el conductismo, los cambios en la conducta son acumulativos.

Las leyes específicas del aprendizaje están condicionadas por el E-R (estímulo- respuesta). Hay dos tipos de condicionamiento: Clásico y Operante.

- **Condicionamiento Clásico:** Pavlov (1849-1936) fue un fisiólogo ruso que ganó un premio nobel por su estudio sobre el funcionamiento de las glándulas digestivas. Trabajó en su laboratorio de forma experimental y controlada con perros, a los que incomunicaba del exterior: su laboratorio se conoció como "las torres del silencio". Durante su experimento Pavlov descubrió una conexión entre el estímulo y la respuesta, ya que notó que cuando en la situación experimental un perro escuchaba las pisadas de la persona que habitualmente venía a alimentarlo, salivaba antes de que se le ofreciera efectivamente la comida; no obstante, si las pisadas eran de un desconocido, el perro no salivaba. Pavlov repitió el experimento para concentrarse en la conducta, esta vez antes de alimentar a los perros hacía sonar un metrónomo (estímulo neutral), e inmediatamente después presentaba la comida al animal (estímulo incondicional). Se repitió esta relación durante varias semanas,

siempre que el perro estuviera hambriento. Transcurridos varios días, hacía sonar el metrónomo y la respuesta de salivación aparecía al oírse el sonido, a pesar de que no se le presentó la comida. Pavlov concluyó que una persona o animal puede ser condicionado para asociar un estímulo neutro con un estímulo significativo, y gradualmente responderá al estímulo neutro de la misma forma que al significado. En el experimento de Pavlov el perro asociaba el sonido de la campana (estímulo neutro) con el alimento (estímulo significativo) y respondía al sonido como si fuera el alimento mismo (salivación).

Se llamó refuerzo, al fortalecimiento de la asociación entre un estímulo incondicional con el condicional, esto es la repetición de los estímulos a lo largo de las semanas. El reforzamiento incrementa la probabilidad de que ocurra determinada respuesta ante ciertos estímulos.

- **Condicionamiento Operante:** también denominado condicionamiento instrumental. El creador de esta teoría es Skinner (1904-1990), quien realizó un experimento en el cual los animales llevaban a cabo una conducta particular y experimentaban una consecuencia. Si la consecuencia era placentera el animal repetía la conducta. Si la consecuencia era desagradable se reducían las probabilidades de que el animal repitiera la conducta. Las consecuencias placenteras se denominan “Recompensas” y las consecuencias desagradables “Castigos”.

Para Skinner la educación en las escuelas falla debido a que las lecciones y exámenes están ideados para revelar lo que no conocen y lo que no pueden hacer los niños en lugar de exponer y construir sobre lo que conocen y pueden aprender. Skinner desarrolló los primeros “programas de aprendizaje” para usarse en máquinas de enseñanza en un intento de aplicar su teoría a la Educación.

1.1.2 El Constructivismo

En este apartado se describirá brevemente el trabajo de dos representantes de esta corriente el epistemólogo, psicólogo y biólogo suizo Jean Piaget (1896-1980) y el psicólogo ruso Lev Vygotsky (1896-1934).

Piaget sostenía que el individuo desarrolla su propia inteligencia y su conocimiento adaptándose. El desarrollo cognitivo es básicamente el conjunto de habilidades matemáticas y lógicas que el niño puede adquirir. Este se produce en cuatro etapas principales:

1. **Sensomotriz:** (del nacimiento - 2 años): el niño llega al mundo con todos sus sentidos y con algunos reflejos para sobrevivir tales como llorar y mamar.
2. **Pre-operacional:** (2-7 años): esta etapa se caracteriza por la descomposición del pensamiento en función de imágenes, símbolos y conceptos.
3. **Operacional concreto:** (7-11 años): el niño ya demuestra un pensamiento lógico ante los objetos físicos.
4. **Operacional formal:** (11-15 años): en este periodo el niño se caracteriza por la habilidad de pensar más allá de la realidad concreta.

Las nuevas experiencias liberan un episodio de “confusión” que desencadena el desarrollo cognitivo debido a que las personas deben adaptar sus viejos conocimientos. Piaget describe dos procesos de Adaptación.

1. **Asimilación:** las nuevas experiencias son reinterpretadas para asimilarlas con las antiguas concepciones. Jean Piaget define esta etapa como la acción del organismo sobre los objetos que lo rodean, en tanto que esta acción depende de las conductas anteriores referidas a los mismos objetos o a otros análogos. En efectos, toda relación entre un ser vivo y su medio presenta ese carácter específico de que el primero (el sujeto), en lugar de someterse pasivamente al segundo (el objeto), lo modifica imponiéndole cierta estructura propia (del sujeto).

2. **Acomodación:** las antiguas concepciones se reestructuran para incluir nuevas experiencias.

Para Piaget la construcción de estructuras se ve afectado por cuatro factores estos son:

1. **La madurez física:** la cual limita su capacidad de aprender y cómo logra hacerlo. El potencial para aprender aumenta a medida que los niños crecen.
2. **La experiencia:** derivada de la acción permanente del sujeto con el mundo físico. Un factor clave en la adaptación, imprescindible para construir estructuras de conocimiento.
3. **Interacción y transmisiones sociales:** referido a las interacciones sociales generales así como también a las transmisiones y formaciones culturales y educativas propias de cada sociedad o medio social en particular.
4. **Autorregulación:** también llamado equilibración, se refiere a la organización interior y progresiva de los conocimientos. El proceso de la autorregulación va de lo simple a lo complejo, parte del control del propio cuerpo hasta el entendimiento y aplicación de las reglas, relacionándolas con las experiencias pasadas y futuras para lograr integrarlas en su diario vivir.

En cuanto al lenguaje, Piaget lo describe como un sistema de símbolos que sirve para representar el mundo como algo diferente de los actos y operaciones que forman los procesos de razonamiento. Por ejemplo, si un niño de cinco años tiene seis manzanas en sus manos y le preguntamos ¿Cuántas tiene? El niño no sabrá responder lógicamente a la pregunta ¿Cuántos? Supongamos que las contamos y le repetimos la pregunta, aunque el niño responda bien según Piaget el niño lo que ha aprendido es un procedimiento (contar y responder una pregunta) y no el concepto de número en sí.

Por su parte Vygotsky definía la inteligencia como la capacidad de aprender por medio de la instrucción. Desarrolló la teoría del “Desarrollo Próximo” referida a la brecha que existe, en el individuo, entre lo que es capaz de hacer solo y lo que puede lograr con la ayuda de alguien experto en un tema determinado. Propone que el momento más significativo en el desarrollo intelectual ocurre cuando el habla y la habilidad práctica confluyen, dando origen a manifestaciones de inteligencia práctica y abstracta. El lenguaje cobraría especial importancia en la medida que se le permite al niño planificar, reduciendo la espontaneidad de su comportamiento y la manipulación directa de las herramientas. Al convertirse en sujetos y objetos de su propio comportamiento con la ayuda del habla, pueden ejercer control sobre él.

1.1.3 La Programación Neurolingüística

La Programación Neurolingüística (PNL) se originó en Santa Cruz, California, durante los años setenta cuando John Grinder (lingüista) y Richard Bandler (psicólogo), preocupados por la excelencia humana, se interesaron en investigar por qué los tratamientos de grandes psiquiatras como: Fritz Perls, Virginia Satir, Milton Ericson y Moshe Feldenkrais, alcanzaban mayor eficacia que el de sus colegas y lograban cambios profundos en sus pacientes. Después de largas investigaciones, y apoyados en la observación sistemática, Grander y Bandler llegaron a la conclusión de que el procedimiento que empleaban los terapeutas era la utilización de un patrón de comunicación muy particular.

Cruz (2005)⁽⁹⁾ sostiene que “la clave estaba en la capacidad del terapeuta para comunicar sus ideas, consejos y recomendaciones en un estilo que armonizaba o concordaba con el mapa mental preferido del paciente” (p. 209)

El nombre de la PNL sugiere que podemos programar nuestros pensamientos y comportamientos; también refleja que hacemos contacto con el mundo, mediante procesos neurológicos a través de los cinco sentidos y utilizamos el lenguaje para comunicarnos. Esto significa que en todas las actividades del ser humano están

relacionadas las palabras, pensamientos y conductas con los objetivos que se pretenden alcanzar.

O'Connor y Seymour (2000)⁽²⁷⁾ plantean que la PNL “es un conjunto de modelos, habilidades y técnicas para pensar y actuar de forma efectiva en el mundo” (p.22). Es posible lograr cualquier meta propuesta y alcanzar la excelencia en las actividades que desarrollemos. Para esto es necesario que pensemos positiva y organizadamente sobre lo que deseamos obtener, trabajar con dedicación y esmero para lograr tal fin.

Por su parte, Sambrano (2005)⁽³⁴⁾ indica que la PNL consiste en una “serie de técnicas destinadas a analizar, codificar y modificar conductas, por medio del estudio del lenguaje, tanto verbal como gestual y corporal” (p. 9). Analizar lo que decimos y la manera de decirlo nos brinda la oportunidad de cambiar aquellos mensajes que entorpecen la comunicación e imposibilitan que las relaciones interpersonales e intrapersonales sean efectivas.

En educación hay varias áreas donde se pueden aplicar sus estrategias para mejorar la relación enseñanza-aprendizaje, la comunicación, aumentar la creatividad, aprender a aprender, aprender a enseñar, mejorar la eficacia del docente, etc.

1.1.3.1. Los Sistemas de Representación

La PNL establece que existen tres modalidades o mapas mentales mediante los cuales las personas perciben e interpretan el mundo que los rodea; es decir, la manera como recogemos, almacenamos y codificamos la información en nuestra mente; estas modalidades se conocen como sistemas de representación (ver Figura I-1). Los neurolingüistas han encontrado que esos mapas son diferentes en cada una de las personas; sin embargo, todas tienen en común variadas modalidades que dependen de los distintos órganos de los sentidos que están involucrados en la manera de tomar contacto con el mundo exterior. La PNL vincula el lenguaje con los sistemas de representación, de tal manera que podemos expresar una misma idea

de cuatro formas diferentes: visual, auditivo, kinestésico y olfativo-gustativo, lo cual implica mirar, escuchar, sentir, olfatear, degustar y contactar.

Estos sistemas de representación son de gran ayuda para el docente pues en la medida que los identifique en sus alumnos y alumnas podrá planificar mejor las actividades escolares ya que “si sabemos cuál es el mapa mental que prefiere utilizar determinada persona para interpretar el mundo exterior, tendremos una herramienta muy útil para saber cuál es la mejor manera de comunicarnos con ella” (Cruz p. 210)

(9)

SISTEMAS DE REPRESENTACIÓN Una misma idea expresada de cuatro formas distintas	
Expresión	Sistema de Representación
No lo veo claro	Visual 
No me suena bien	Auditivo 
No agarro la idea	Kinéstesico 
No termino de tomarle el gusto	Olfativo- gustativo 

Figura I-1. Sistemas de Representación. Fuente: Páez v Sansonetti 2011.

Para tener una comunicación efectiva, es importante escuchar con atención las palabras que pronuncian nuestros interlocutores, pues de esta manera podremos identificar qué tipo de sistema de representación utilizan; además, estaremos en capacidad para comprenderlos y crear una atmósfera favorable al intercambio de ideas.

1.1.3.2 Las Señales de Acceso Ocular

Otra estrategia que plantea la PNL y que permite identificar la forma cómo piensa una persona, son las señales de acceso ocular. Estas señales se refieren a las diferentes maneras de mover los ojos cuando pensamos. Permiten identificar de

qué manera buscamos la información que está almacenada en nuestro cerebro y, además, brindan la oportunidad de saber si una persona piensa mediante imágenes, sonidos o sensaciones. Este es un proceso completamente inconsciente, pues activamos aspectos neurológicos que reflejamos con los movimientos. Con respecto a este punto, Sambrano (2005)⁽³⁴⁾ plantea:

Todas las personas presentan una conexión neurológica innata que los hace mover los ojos hacia diversas direcciones, de acuerdo con el sistema representativo que en ese momento están invocando en su pensamiento; este es un proceso completamente reflejo e inconsciente y es muy difícil manejarlo en forma deliberada porque obedece a la búsqueda de redes neurales que poseen la información que la persona está buscando en un momento determinado. (p. 48)

El componente visual: se refiere a la manera en que buscamos la información, según enfoquemos la mirada. Cuando miramos hacia arriba y a la derecha estamos creando. Esto permite detectar cuando una persona está mintiendo o inventando una excusa. Si miramos hacia arriba y a la izquierda, estamos recordando datos almacenados en nuestra mente (ver Figura I-2).



Figura I-2. Componente Visual. Fuente: metodoprofesional.es 2011

El componente auditivo: indica que cuando movemos los ojos horizontalmente abrimos nuestro canal auditivo, lo cual implica que creamos sonidos si miramos hacia la derecha, mientras que si miramos a la izquierda, estamos recordándolos (ver Figura I-3).



Figura I-3. Componente Auditivo. Fuente: metodoprofesional.es 2011

El componente kinestésico: Al mirar hacia abajo y a la izquierda establecemos un diálogo interno; abajo y a la derecha, significa que tomamos conciencia de nuestro cuerpo y también que experimentamos las emociones en el cerebro. Cuando enfocamos los ojos y miramos hacia delante sin objetivo aparente, es posible que estemos visualizando y por último, cuando miramos hacia abajo y al centro: el patrón visual indica confusión (ver Figura I-4).



Figura I-4. Componente Kinestésico. Fuente: metodoprofesional.es 2011

Es importante que el docente desarrolle agudeza sensorial para identificar la manera que tienen sus alumnos y alumnas de expresarse a través de los procesos mentales mencionados, pues esto le brinda la oportunidad de diseñar estrategias de enseñanza y modificar el lenguaje utilizado en las actividades escolares. Estas actividades deben estar acordes con los tres mapas mentales ya que las personas visuales ven el mundo a través de la formación de imágenes; las auditivas diseñan su respuesta basándose en lo que escuchan más que en lo que ven; y las personas kinestésicas actúan guiadas por lo que sienten y por las sensaciones que experimentan.

Aplicando las estrategias que ofrece la PNL se logra mejorar, considerablemente, la interacción en el aula; al mismo tiempo, se suministran herramientas para que los alumnos y alumnas alcancen un aprendizaje significativo, lo cual incluye la creatividad.

1.1.4 La Teoría Uno

David Perkins (1995)⁽³¹⁾ afirma que "la gente aprende más cuando tiene una oportunidad razonable y una motivación para hacerlo". Esta teoría según su autor es la teoría del aprendizaje basada en el sentido común, siendo factores importantes en la enseñanza:

- **La información clara:** se describen de forma precisa los objetivos establecidos, los conocimientos requeridos y los resultados esperados.
- **Práctica reflexiva:** se conceden oportunidades a los aprendices o usuarios (en el caso de aplicaciones para dispositivos móviles) de ocuparse activa y reflexivamente de aquello que deba aprender
- **Realimentación informativa:** se proporcionan consejos claros y precisos para que el alumno mejore el rendimiento y pueda proceder de manera más eficaz.
- **Fuerte motivación intrínseca y extrínseca:** las actividades son ampliamente recompensadas.

La Teoría Uno es compatible con el conductismo y con el constructivismo, pero no enfatiza la importancia de que el alumno elabore sus ideas con un alto grado de autonomía a fin de alcanzar la verdadera comprensión.

Perkins explica que existe una correspondencia entre la comprensión y las imágenes mentales, por lo que se puede decir que es una relación es bilateral. El aprendiz puede adquirir imágenes mentales, con lo cual desarrolla su capacidad de comprensión y al exigirles actividades de comprensión (como por ejemplo: predecir, explicar, resolver, ejemplificar, generalizar) se propiciará la construcción de imágenes mentales.

1.1.5 Teoría Cognitiva del Aprendizaje Multimedia

Mayer y Moreno proponen una teoría cognitiva del aprendizaje multimedia basada en los siguientes supuestos:

- Las “reservas” de memoria a corto plazo para estímulos visuales y auditivos son independientes.
- Cada memoria a corto plazo tiene una capacidad limitada.
- Los humanos poseen sistemas separados para representar la información verbal y la no verbal.
- El aprendizaje significativo ocurre cuando un estudiante selecciona información relevante en cada reserva, organiza la información en cada reserva en una representación coherente y crea conexiones entre esas representaciones.

A partir de una serie de experiencias realizadas con alumnos universitarios, los autores proponen una serie de principios que mejoran la enseñanza multimedia:

- **Principio de modalidad:** el discurso debe presentarse de manera hablada más que como texto en pantalla.
- **Principio de contigüidad temporal:** El principio de contigüidad temporal significa que las palabras y las imágenes correspondientes se deben presentar al mismo tiempo; los estudiantes aprenden más cuando el material verbal y visual está sincronizado temporalmente que si estos se presentasen en distintos tiempos separados entre sí.
- **Principio multimedia:** se debe emplear de forma simultánea imágenes, animaciones y palabras ya que cuando se presentan juntas el discurso

hablado y las imágenes, los participantes tienen la ocasión de construir representaciones cognoscitivas verbales y visuales e integrarlas.

- **Principio de personalización:** sugiere que el texto y/o la narración debe hacerse en primera o segunda persona más que un estilo formal de tercera persona.
- **Principio de coherencia:** debe excluirse cualquier material extraño al contenido que se quiere enseñar. Todo lo que fuese “más entretenido” pero no tiene relación directa con el tema debe desecharse.
- **Principio de señalización:** este principio indica que ocurre una mejor transferencia cuando se señala o dirige el recorrido visual hacia el elemento clave. El señalar asiste a los aprendices en el proceso de organizar la información, lo que puede dar lugar a un aprendizaje más profundo y significativo.
- **Principio de establecimiento de ritmo:** el ritmo de la presentación es de acuerdo con la audiencia, más que por el programa o el expositor. El público puede variar el tiempo necesario para, enganchar los procesos cognoscitivos de selección, organización e integración de la información entrante, así que la retroalimentación marca el ritmo para regresar o detener brevemente la presentación en caso de ser necesario. Si el ritmo actual del material es demasiado rápido, puede que estos procesos cognoscitivos no sean ejecutados correctamente y los miembros del público se pierdan.

1.2 Características evolutivas de los niños según su edad

Hay una serie de características particulares en los niños de determinadas edades, que deben tomarse en cuenta al momento de diseñar un software educativo.

1.2.1 El niño de tres años

- Camina y corre, pero todavía carece de coordinación.
- Le gusta hacer cosas con sus propias manos pero lo hace con cierta torpeza.
- Tiene una mayor capacidad para el lenguaje. Le agrada hablar y aprender nuevas palabras. Su atención es de corto alcance. Es curioso e inquisitivo. Con frecuencia no entiende bien y suele hacer comentarios aparentemente ajenos al tema. Le agrada usar su imaginación. Le gustan los juegos con los dedos y las manos, los cuentos y las actividades musicales. No alcanza a diferenciar la fantasía de la realidad.
- Le agrada trabajar a solas. No le interesa jugar en cooperación con otros, pero le gusta tener amigos a su alrededor. Es egocéntrico. Tiene dificultad en compartir. Prefiere estar cerca de personas adultas, particularmente familiares, porque así se siente protegido.
- Le agrada complacer a las personas adultas. Necesita de su aprobación, su amor y sus elogios. Reacciona emocionalmente cuando siente temor o ansiedad. Lloro con facilidad. Es sensible a los sentimientos de otras personas. Está desarrollando cierta independencia. Experimenta emociones intensas, aunque breves.

1.2.2 El niño de cuatro años

- Es muy activo. Se mueve con rapidez. Le gusta brincar, saltar, correr, trepar y arrojar cosas.
- Le agrada conversar y aprender nuevas palabras. Hace muchas preguntas. Alcanza a razonar un poco, pero todavía tiene muchas ideas equivocadas. Tiene dificultad en diferenciar la realidad de la fantasía. Su atención es de corto alcance. Emplea dibujos para expresar sus sentimientos. Disfruta de usar su imaginación y de las representaciones dramáticas.

- Juega con mayor espíritu de cooperación con otros.
- Suele ser físicamente agresivo, dominador, grosero y obstinado, pero puede asimismo ser amigable. Está aprendiendo a compartir, a aceptar las reglas y a tomar su turno. Reacciona ante todo elogio sincero.
- Con frecuencia, trata de ver hasta dónde podrá salirse con la suya. Es jactancioso, especialmente en cuanto a sí mismo y a su familia. Podría ser complaciente en un momento y pendenciero en el siguiente. Tiene más confianza en sí mismo. Podría tener ciertos temores y sentimientos de inseguridad.

1.2.3 El niño de cinco años

- Es muy activo. Tiene buen sentido del equilibrio y está llegando a tener mejor coordinación. Puede patear una pelota, caminar en línea recta, saltar, brincar y marchar. Le agrada dibujar, colorear y participar en actividades y en juegos; está aprendiendo a ponerse los zapatos y a atarse los cordones de los zapatos, y a abotonarse la ropa.
- Reconoce algunas letras, cifras y palabras. Le gusta aparentar que sabe leer y escribir. Podría estar aprendiendo a leer. Es conversador. Hace preguntas y comentarios, y ofrece contestaciones que demuestran un mayor entendimiento. Sabe resolver ciertos problemas. Es curioso y anhela conocer los hechos. Está comenzando a distinguir entre la verdad y la fantasía. Su atención es de corto alcance pero está empezando a incrementarse. Prefiere las tareas concretas. Le gustan los acertijos y las bromas, pero no puede reírse de sí mismo. Le agradan los cuentos, las canciones, las poesías y las dramatizaciones.
- Es amigable y está ansioso por complacer y cooperar. Está comenzando a gustarle encontrarse en pequeños grupos de niños, pero podría preferir tener

un mejor amigo. Provoca menos conflictos al jugar en grupos. Está empezando a querer amoldarse y critica a los que no desean hacerlo. Está comenzando a entender las reglas, pero suele tratar de cambiarlas para su propio beneficio.

1.2.4 El niño de seis años

- Es muy activo. Suele ser ruidoso, inquieto y vivaz. Le agrada participar en actividades y llevar a cabo pequeñas tareas, aun cuando pudieran ser difíciles de realizar. Le desagrada ser simplemente un espectador.

- Necesita que se le enseñen conceptos en maneras específicas. Está mejorando su memoria. Es conversador y hace muchas preguntas. Está aprendiendo a tomar decisiones, pero suele ser indeciso. Está incrementando su tendencia a prestar atención. Le gusta leer, escribir, cantar, escuchar cuentos y usar su imaginación.

- Está más interesado en actividades en grupo y en relacionarse con compañeros de juego, pero continúa siendo un tanto egoísta. En ocasiones suele ser dominador, agresivo o descortés con sus compañeros. Tiene amistades inestables. Se preocupa por la forma en que otros lo tratan. Anhela ser aprobado socialmente.

1.3 Métodos de enseñanza de la lectura

La lectura es la interacción entre un texto escrito y un lector, en el cual se producen una serie de intercambios a través del lenguaje, expresado en signos y códigos convencionales que nos sirven para interpretar dicho texto. Existen numerosos métodos para la enseñanza de la lectura, organizados principalmente en tres grupos:

1.3.1 Método Sintético o Silábico

Este Método consiste en tomar como base los elementos mínimos del lenguaje escrito; letras, signos y sílabas y, una vez memorizadas, fusionarlas en otras estructuras más complejas hasta llegar a las palabras, frases o textos. Este método se divide en tres:

- **Alfabético:** conocido también como método lineal, es el más antiguo de todos, se remonta a las épocas antes de Cristo. Se trabaja la forma y el nombre de cada una de las letras de manera independiente para, después, combinarlas creando sílabas y palabras. Se empieza con las vocales, en el orden “i”, “o”, “a”, “e”, “u”, se sigue con las consonantes, generalmente con la “p” o la “m”, que resultan más fáciles para los niños, y con ellas se van formando las primeras sílabas y palabras.
- **Fonético:** parte del sonido de la consonante como unidad mínima. Se le enseña al niño el sonido de cada letra en particular no por su nombre. Ejemplo: la letra “m” no se pronuncia “eme” sino “em”. Los defensores de este método consideran que ayuda a los niños a comprender más fácilmente la relación entre fonema y letra, y a articular las palabras de forma correcta y sin errores.
- **Silábico:** derivado del fonético y el alfabético, en este método se enseñan las vocales (dentro de palabras con ilustraciones y palabras) por ejemplo: u de uva para luego componer sílabas. Le permite a los niños captar mejor este sonido y aprender a leer con más facilidad.

1.3.2 Método Analítico o Global

Este método parte de la palabra o la frase para llegar a sus elementos: las sílabas, las letras y los sonidos apoyándose en las imágenes. Los especialistas describen este método como “natural” e “intuitivo”. Este método se divide en dos:

- **Global:** la palabra se percibe como un todo con una forma asociada a un significado. Sugiere que los niños deben aprender a leer palabras completas,

para llegar a sus elementos: las sílabas, las letras, los sonidos. Para ello se deben colocar en los entornos en que se desenvuelve el niño (la casa y/o escuela) rótulos con los nombres de los objetos, en letras grandes y de molde, en un lugar visible de manera que los niños puedan recordarla. Ejemplo: una puerta con el rótulo “PUERTA” pegado en él.

Tras muchas repeticiones, los niños aprenden a leer frases o textos con las palabras aprendidas. Este método sostiene que los niños por sí mismos descubren la relación letra- sonido así como el resto de los elementos de la oración, como los verbos. Los defensores de este método sostienen que es un método más natural, porque sigue el mismo proceso que el aprendizaje del lenguaje, que se basa en repetir las palabras que oímos continuamente.

- **Modelo Carteles de Experiencia:** se basa en crear situaciones de aprendizaje a partir de las experiencias individuales. El aprendizaje se dará de las relaciones que el niño experimente con la lengua escrita. Con este modelo se favorece el pensamiento, logra diferenciar los dibujos de la escritura, dando paso al análisis y codificación del lenguaje escrito.

1.3.3 Métodos Mixtos

También conocido como ecléctico. Como su nombre lo sugiere es una composición de los métodos descritos anteriormente (sintético y analítico). Sostiene que las estrategias para la enseñanza de los procesos de la lectura y la escritura deben hacerse de forma combinada de acuerdo al desarrollo y necesidades de cada niño en particular.

Según Uhia A. para la enseñanza de la lectura es fundamental tener en cuenta un orden lógico, de esta manera se hace más fácil y organizado; se debe comenzar por lo sintético donde es importante que el niño reconozca las palabras y sonidos; y los analíticos que son los elementos fundamentales para darle significado a la lectura.

Cada método descrito anteriormente tiene sus ventajas y desventajas. Para este trabajo de investigación se considera oportuno utilizar el método mixto ya que atiende las diversas formas de aprendizajes de los niños.

1.4 Teorías sobre el diseño de software educativo

Según Pere Marqués los términos software educativo, programas educativos y programas didácticos son sinónimos. Define como Software Educativo: “a los programas para computadoras creados con la finalidad específica de ser utilizados como medio didáctico, es decir, para facilitar los procesos de enseñanza y de aprendizaje” [\(24\)](#).

Galvis Panqueva denomina “software educativo a aquellos programas que permiten cumplir o apoyar funciones educativas”.

Existen varias metodologías para normar el desarrollo de un Software Educativo, entre ellas: la Metodología para el Desarrollo de Software Educativo (DESED), el Modelo Sistemático, los Modelos no lineales y los Modelos de Hipertexto. Cabe destacar que estas metodologías trabajan en conjunto con los métodos tradicionales de desarrollo de software (Métodos Ágiles, *RAD*, *XP*, *Cascada*, etc.) y son importantes a la hora de desarrollar softwares para uso educativo.

1.4.1 Metodología para el Desarrollo de Software Educativo (DESED)

Esta metodología sostiene que el desarrollador del software educativo (SE en adelante) planea su producto considerando las características planteadas en cada fase del desarrollo; la finalidad es la creación de productos de softwares creativos, que se adapten a las necesidades del usuario final considerando su edad, el medio ambiente, el contenido, entre otros.

Esta metodología consta de 13 pasos fundamentales, en los cuales se toman en consideración aspectos de Ingeniería de Software, Educación, Didáctica y Diseño gráfico, entre otros (ver [Figura I-5](#)); estos pasos son:

1. Determinar la necesidad de un SE: el SE deberá cubrir los aspectos primordiales del área de estudio de que se trate, y que la necesidad de desarrollar un producto debe permitir al Ingeniero de Software hacerse de la información y las técnicas didácticas que pudieran ser empleadas al impartir normalmente la asignatura. Además, debe mejorar sustancialmente la calidad de la educación.

2. Formación del equipo de trabajo: diversos autores analizados concuerdan en que se requiere conformar un grupo de trabajo nutrido para poder desarrollar un SE completo, esto debido a que lo más importante ya no es sólo la información, sino que también debe tenerse muy presente la forma de presentar la información, que en un momento dado se convierte en conocimiento que debe ser adquirido por los estudiantes.

3. Análisis y delimitación del tema: es el momento de reunir la información obtenida y definir la extensión del SE. Se analizan las necesidades presentadas por las personas que requieren el software, determinándose los objetivos particulares de trabajo, es decir, las necesidades deben permitir establecer el ámbito de la materia, y determinar los temas específicos, de los planes de estudio, que deben ser considerados para el desarrollo del producto; y esto es sumamente importante, ya que se debe delimitar la amplitud de los temas a cubrir.

4. Definición del usuario: deben determinarse las características del usuario, la edad será determinante para la elección y aplicación de las técnicas de enseñanza que se vayan a tener presentes en el desarrollo del software.

5. Estructuración del contenido: se deben definir los conceptos a considerar para establecer los contenidos temáticos que se abarcan en el SE. El trabajo conjunto entre los expertos en el tema y el diseñador de software

6. Elección del tipo de software a desarrollar: en esta etapa se deben establecer los niveles de complejidad de las áreas de aprendizaje, así como también las técnicas de aplicación implícitas o explícitas; ejercitación y práctica, simulación, tutorial; uso individual, competición.

7. Diseño de interfaces: la interfaz es lo que contribuirá a la motivación, eficiencia, comprensión y uso del SE que se desarrollará. Se toman en cuenta las consideraciones didácticas expuestas en la definición de necesidades.

8. Definición de las estructuras de evaluación: debe proporcionarse a la par de los contenidos de aprendizaje, las formas de evaluación de los contenidos mismos.

9. Elección del ambiente de desarrollo: se debe delimitar el campo de aplicación del SE, ya que cada desarrollador deberá buscar la herramienta que le permita involucrar todas las peticiones de los usuarios potenciales.

10. Creación de una versión inicial: una vez que se tiene la información requerida del índice temático, que se ha elegido el ambiente de desarrollo y el tipo de software a realizar, se debe comenzar a planificar los aspectos de implementación y realizar la ejecución en sí.

11. Prueba de campo: la primera versión del sistema debe ser puesta a prueba frente al equipo de trabajo para su evaluación y rectificación de características. Si se detectan errores u omisiones, debe retomarse el desarrollo y volver a orientar la implementación del nuevo diseño de las modificaciones realizadas, creando una nueva versión del SE.

12. Mercadotecnia: en el caso de que el SE haya sido diseñado para comercializarlo, debe hacerse un recuento de características de mercadotecnia que harán que el producto sea vendible. Debe elegirse un nombre, un empaque, el modo de distribución. La estrategia de mercado elegida, es la que hará el software

incursione y se presente ante los usuarios finales potenciales, para que pueda afianzarse un mercado.

13. Entrega del producto final: debe presentarse un producto final a los usuarios potenciales, el cual debe tener el apoyo documentado en características de instalación, operación.

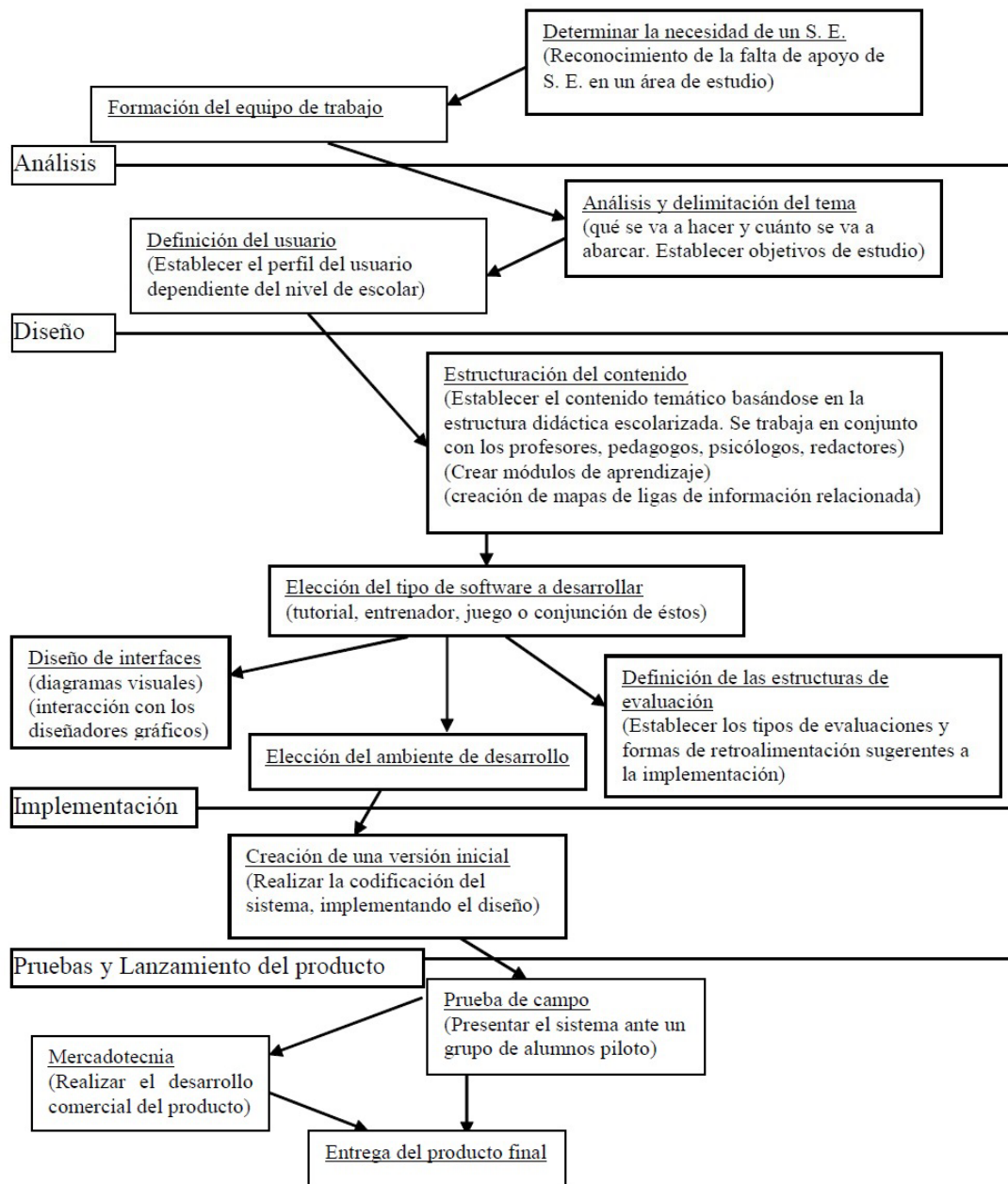


Figura I-5. Esquema general de la Metodología DESED. Fuente: Peláez y López 2006.

1.4.2 El Modelo Sistemático.

Se origina en la Ingeniería del Software y fue adaptado a la producción de software educativo. Álvaro Galvis considera la elaboración del software como un

proceso en orden lineal. Este modelo está formado por cinco fases independientes, las cuales son: análisis, diseño, desarrollo, implementación y evaluación (ver figura I-6). El punto de partida de la metodología es la identificación de necesidades educativas reales que conviene atender con Material Educativo Computarizado (MEC). La prueba piloto es la base para decidir si el MEC debe llevarse a la práctica en gran escala, o debe rediseñarse, ajustarse o desecharse.

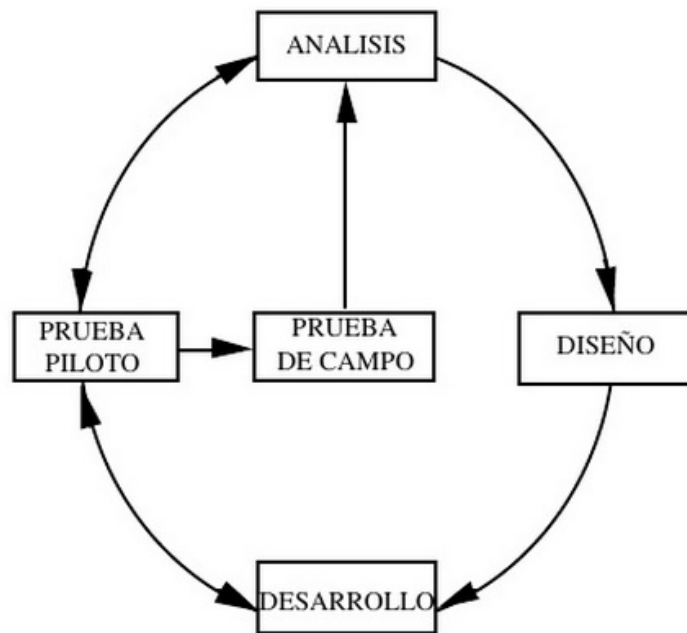


Figura I-6. Modelo Sistemático para la selección o desarrollo de MEC. Fuente: Galvis 1992.

Durante su implementación también es importante que se evalúe el MEC de modo que se pueda establecer la efectividad real del material, por ello es importante realizar pruebas de campos.

1.4.3 Modelos no lineales

Estos modelos permiten la continua revisión y actualización del producto. Existen numerosos modelos no lineales del desarrollo de software educativo, uno de los más reconocido es el denominado Metodología THALES, propuesto por Madueño

(2003) cuya idea central se basa en el diseño y desarrollo del software educativo como un proceso de resolución de problemas. Está formado por cinco fases, a saber: formulación de los objetivos, diseño del programa, prototipos (soluciones), revisión de las soluciones y revisión de los objetivos (ver Figura I-7).

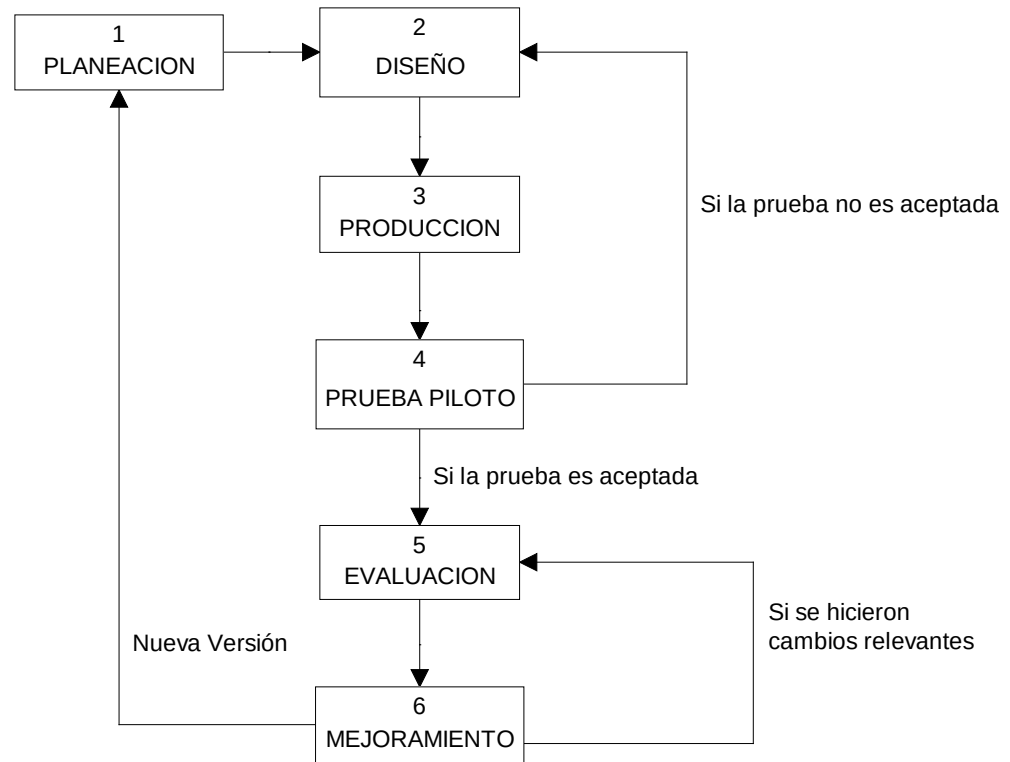


Figura I-7. Metodología Thales. Fuente: Madueño 2001.

En este sentido los lenguajes de autor (lenguajes orientados al desarrollo de archivos o documentos multimedia) han facilitado la aplicación de este modelo, ya que estos permiten ejecutar el programa aunque no esté completamente terminado.

Otros modelos no lineales son los llamados modelos de hipertexto, los cuales son sistemas de administración de Base de Datos que permiten conectar pantallas de información, utilizando enlaces asociativos. Tienden a imitar la habilidad del cerebro para almacenar y recuperar información mediante enlaces referenciales para un acceso rápido e intuitivo.

CAPÍTULO II:

TRABAJOS PREVIOS

A fin de apoyar la fundamentación teórica de este trabajo especial de grado, se hizo necesario la revisión exhaustiva de trabajos de investigación previos realizados tanto a nivel nacional como internacional, desarrollados en los países de: España, Estados Unidos, México y Venezuela: cuya línea de investigación está relacionada con las aplicaciones móviles, la mayoría de ellos, con fines educativos.

Dicha revisión se presenta a continuación, son seis trabajos que reportan hallazgos interesantes sobre el alcance de las aplicaciones móviles utilizadas como apoyo a los procesos de enseñanza y aprendizaje.

2.1 Trabajo: “Análisis de las Características y uso de software educativo para niños en edad Inicial”

Sánchez María Alejandra (2002)⁽³⁶⁾ realizó un trabajo de Investigación titulado “Análisis de las Características y uso de software educativo para niños en edad Inicial”. Esta investigación es de tipo descriptiva, bajo la modalidad Estudio de Campo.

Sánchez deseaba saber si los softwares que se estaban utilizando en las Instituciones de Educación Inicial estaban acorde al Currículo de Preescolar de Venezuela, así como también conocer las características y el impacto de dichos softwares en el aprendizaje de los niños en edad preescolar.

El objetivo general de este Trabajo Especial de Grado fue Analizar las características y uso del software educativo para niños de Educación Inicial. Sánchez

se planteó como objetivos específicos: Investigar de qué forma se combina el uso de la tecnología con la práctica pedagógica, cuál es el software más utilizado por los Centros de Educación Inicial y observar la interacción de los niños con dicho software.

Sánchez visitó ocho Centros de Educación Inicial en Caracas que integrasen la tecnología como práctica pedagógica, encontrando veinte softwares educativos comunes entre estos. Tomó como muestra tres softwares: un software para desarrollar el lenguaje (El camino a la lectura 1), un segundo software para aprender matemáticas (Trampolín Escuela Infantil) y un tercer software para aprender el idioma inglés (*Max comes to Earth*).

Como instrumentos de recolección de los datos se utilizó la técnica de la entrevista al personal docente de cada una de las ocho Instituciones visitadas. Para evaluar las características de cada software se utilizaron listas de cotejo y tablas de verificación.

Sánchez recomienda la constante actualización docente porque no todas las maestras de Preescolar estaban familiarizadas con el computador y la integración de este como practica pedagógica. En cuanto a los softwares estudiados concluyó que se adaptaban al Currículo venezolano de Educación Inicial; que estos eran un refuerzo de las lecciones impartidas en el aula de la clase además de ser un elemento motivador del aprendizaje ya que los niños se mostraban interesados en utilizar los dispositivos electrónicos. Adicionalmente la interacción de los niños con la computadora les permite desarrollar destrezas en el manejo de la misma; en esta época denominada era tecnológica son conocimientos básicos necesarios en el tiempo presente y más aún en su futuro profesional.

2.2 Trabajo: “Süchiki Walekerü: un ejemplo del uso de las TIC en escuelas indígenas. Caso Wayuu”

Quero y Madueño (2006)⁽³²⁾ titularon su trabajo “Süchiki Walekerü: un ejemplo del uso de las TIC en escuelas indígenas. Caso Wayuu”.

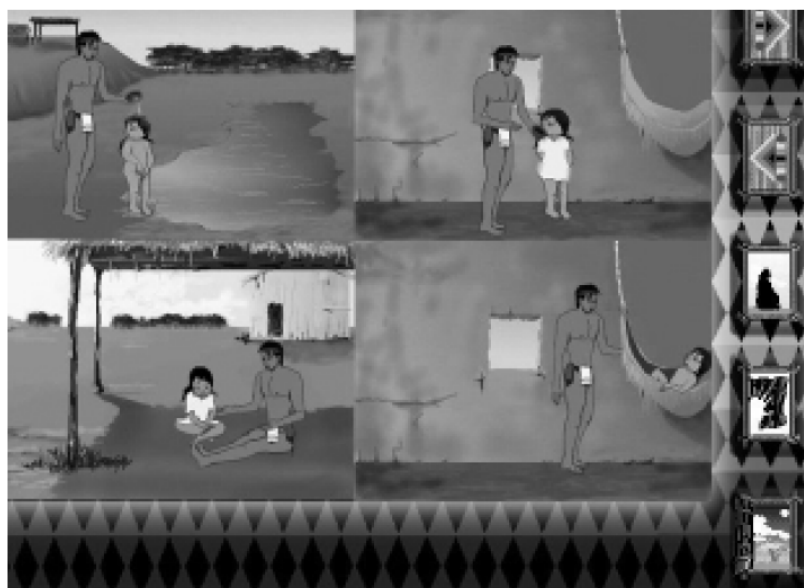


Figura II-1. Impresión de pantalla 1 del software Süchiki Walekerü.

La problemática que plantean estos investigadores es la pérdida del idioma wayuunaiki en la comunidad indígena Wayuu quienes representan un 57% de la población total indígena en Venezuela según el Censo Nacional Indígena del año 2000; ya que aunque la mayoría de sus miembros son hablantes del mismo, no lo leen ni lo escriben. Los investigadores se plantearon como objetivos el diseño, aplicación y evaluación de un software para aprender el Idioma Wayuu, el cual lleva por nombre Süchiki Walekerü; se desarrolló utilizando la metodología de software educativo Thales.

Esta investigación tiene un enfoque etnográfico introspectivo–vivencial y la rige el método de investigación–acción (ver Figuras II-1 y II-2). La aplicación de la misma se delimitó a la Escuela Básica Nacional Puerto Aléramo, municipio Páez del estado Zulia, la cual posee un laboratorio de computación donde se pudo instalar el software.



Figura II-2. Impresión de pantalla 2 del software Süchiki Walekerü.

Como conclusiones de su trabajo Quero y Madueño exponen que en general los niños se interesan por el uso del computador (y dispositivos electrónicos), y pese a que el juego es de un idioma desconocido para ellos, no mostraron descontento alguno por el mismo sino que se sintieron identificados al escuchar su lengua ancestral wayuunaiki. Se evidenció que a los niños les gusta leer palabras o frases cortas y escribir poco. Resaltan que para los niños de cortas edades las frases tienen algún sentido si son escuchadas, las palabras escritas no les proporcionan mayor información.

El desarrollo, aplicación y evaluación del software generó un acuerdo entre la Zona Educativa del Estado Zulia (Ministerio de Educación), Coordinación de Educación Intercultural Bilingüe, UNICEF y los investigadores responsables del desarrollo del mismo. Dicho acuerdo consiste en la ejecución de talleres para la aplicación del software y las estrategias didácticas a seguir con todos los docentes de la escuela Puerto Aléramo y tres escuelas adicionales del municipio Páez del Estado Zulia.

2.3 Trabajo: “El aprendizaje de las matemáticas en segundo grado de primaria por medio de dispositivos móviles”

Aquino A. Zahedi (2007)⁽¹⁾ realizó una investigación denominada “El aprendizaje de las matemáticas en segundo grado de primaria por medio de dispositivos móviles” cuya finalidad era diseñar un software educativo utilizando dispositivos Palm (dispositivo móvil), para niños de segundo grado de Educación Básica con edades comprendidas entre los siete y ocho años de edad como apoyo a la enseñanza de las matemáticas en el aula.



Figura II-3. Impresión de pantalla 1 del software de Aquino A. Zahedi.

Dicho software incluiría temas en los cuales los niños presentasen mayor dificultad para aprender. Estos temas serían el resultado de encuestas a los docentes de segundo grado. Los niños serían encuestados sobre colores de su preferencia, esto para desarrollar las pantallas de los juegos de acuerdo a los principales colores en común. Para ello Aquino visitó de ocho escuelas primarias de Huajuapán de León en México.

Las actividades implementadas en el juego se fundamentaron en las actividades que se encuentran actualmente en los libros gratuitos de matemáticas del segundo grado de primaria proporcionados por la Secretaría de Educación Pública de México (ver Figuras II-3 y II-4).

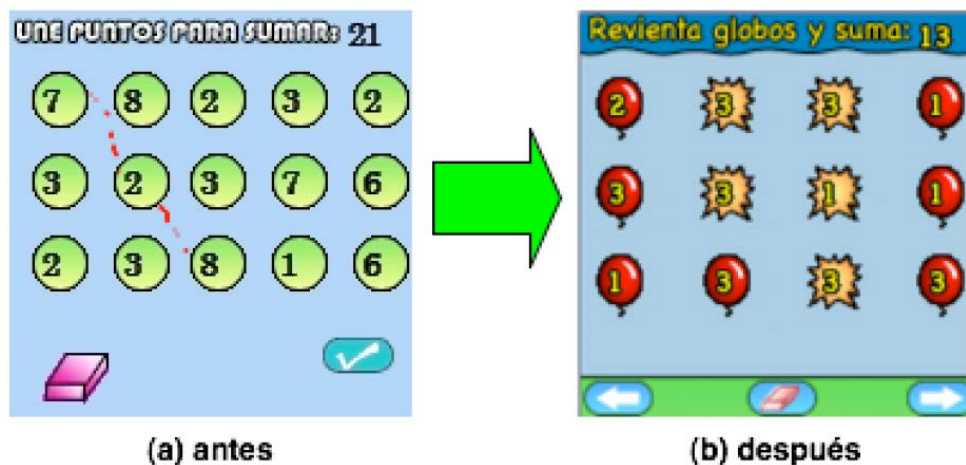


Figura II-4. Impresión de pantalla 2 del software de Aquino A. Zahedi.

La aplicación se desarrolló siguiendo el proceso de Edumóvil que se basa en las metodologías *UCD* (*User Centered Design*) y *GUP* (*Game Unified Process*) en conjunto.

Aquino concluye que en aplicaciones móviles el tamaño y peso del celular es importante ya que esto facilita la manipulación del mismo con una sola mano de manera que puedan utilizarlo con la otra mano. La aplicación desarrollada es extensiva no limitativa ya que es una herramienta de apoyo para los maestros y de repaso para niños de cualquier grado de primaria.

Aquino resalta la importancia de la revisión y mejoras constantes de las *apps* para que así se adapten a las necesidades reales de los usuarios finales.

2.4 Trabajo: “Juegos educativos en dispositivos móviles”

Villoria, Martín y de las Cuevas (2009)⁽⁴⁰⁾ denominaron su trabajo de Grado “Juegos educativos en dispositivos móviles”. La problemática que motiva esta investigación es la inexistencia de fórmulas pedagógicas aceptadas en la comunidad científica que permitan aprovechar eficazmente el potencial de las tecnologías en la educación; por ello los autores deseaban proporcionar una herramienta de autoría

que facilitara la creación de videojuegos educativos para distintos entornos y plataformas, con especial interés en la compatibilidad con dispositivos móviles.

Como objetivo de investigación los autores se plantean proporcionar soporte a la creación de juegos *eAdventure* (una herramienta gráfica para el desarrollo de juegos para *Android*).



Figura II-5. Impresión de pantalla 1 del software de Villoria, Martín y de las Cuevas.

Villoria, Martín y de las Cuevas estudiaron el estado del mercado de los dispositivos móviles con el fin de determinar las plataformas más relevantes. En primer lugar se situó el *iPhone* como la plataforma más extendida abarcando la mitad de todos los dispositivos de la muestra. El segundo lugar lo ocupó el sistema operativo *Android* que está presente prácticamente en uno de cada cuatro dispositivos analizados. Por último se situó *Symbian* usado por el 18% de los dispositivos.

Estos autores utilizaron herramientas colaborativas tales como *Google Wave* (*TaskBoard Widget*) y correo electrónico, *Google Code*, *Google Code Wiki*, *Google Docs*, *Microsoft Word* y *DropBox*, para ayudarles en su trabajo.

Durante el transcurso del proyecto los autores decidieron realizar un prototipo de juego educativo que hiciese uso del sistema de geo-posicionamiento, orientado a motivar el aprendizaje en el lugar y el momento (*Just-in-time learning*). El juego consta de tres capítulos correspondientes a los tres edificios emblemáticos más visitados de Madrid. El jugador deberá resolver los diferentes problemas que le serán planteados mediante la interacción con los personajes y objetos disponibles, al mismo tiempo que resuelve éstos problemas, se le informarán datos históricos y anécdotas que estén directamente relacionadas con la escena en la que se encuentra (ver Figuras II-5 y II-6).



Figura II-6. Impresión de pantalla 2 del software de Villoria, Martín y de las Cuevas.

Finalizada su investigación Villoria, Martín y de las Cuevas afirman que *Android* fue considerada la plataforma móvil más adecuada para el desarrollo del proyecto, ya que permite la libre distribución del contenido y cuenta con un gran número de dispositivos móviles en el mercado. También afirman haber desarrollado una herramienta que facilita la creación y distribución de aventuras gráficas educativas en móviles *Android* de posición global del jugador y la de interacción con el entorno real a través del uso de la cámara de fotos y los códigos bidimensionales. En la parte de edición de juego mantuvieron el mismo proceso con el que contaba la plataforma *eAdventure*. Los investigadores añadieron soporte para perfiles de exportación a *Android* en la que se realiza una comprobación de compatibilidad de

los recursos utilizados en el juego para correr en *Android*. De esta manera el desarrollador puede desplegar los juegos que genere en plataforma de escritorio, *web* o móvil *Android*. Se mantuvo la compatibilidad con los juegos creados en versiones previas por lo que se pueden editar y exportar en móviles sin necesidad de modificaciones.

2.5 Trabajo: “Los Efectos de la Tecnología Multimedia sobre el Aprendizaje”.

Ryan T. Gertner (2011)⁽¹⁵⁾ denominó su trabajo de investigación “Los Efectos de la Tecnología Multimedia sobre el Aprendizaje” cuyo objetivo de investigación se centró en la efectividad de los estudiantes que usan *iPads* con una aplicación de libros electrónicos frente a los estudiantes que utilizan los libros de texto y sus efectos sobre la comprensión y la transferencia de aprendizajes.

Gertner manejó la hipótesis de que los participantes en el grupo del *iPad* obtendrían puntuaciones más altas en la transferencia y en la comprensión en comparación con los participantes en el grupo de los libros de texto.

El estudio contó con sesenta y nueve participantes los cuales fueron reclutados de un curso introductorio de Psicología en la Universidad Cristiana de Abilene del Estado de Texas en los Estados Unidos. Estuvo conformado por 59,4% mujeres y 40,6% varones, de edades comprendidas entre los 17 a los 35 años de edad; de los cuales 69,6% de los participantes eran de raza blanca, el 13% eran hispanos, el 5,8% eran afroamericanos, y el 11,6% eran asiáticos o asiático-americanos.

Como material de prueba para el estudio antes mencionado se utilizó en su forma impresa y electrónica, el libro “*King’s (2010) Experience Psychology*”, Capítulo 11, páginas 393-399. El contenido del capítulo se compone de temas centrados en la psicología social, específicamente de la conformidad, la obediencia, y la influencia de los grupos.

Los participantes fueron divididos en dos grupos, un grupo de lectura en físico y otro grupo de lectura en electrónico; los mismos fueron seleccionados de forma aleatoria y se les otorgó un tiempo de 25 minutos para leer. El grupo que fue asignado a la copia de texto tradicional se les dio una copia a color del texto. Al grupo que le fue asignada la lectura electrónica le fueron entregados *iPads* que ya tenían la sección correspondiente en el libro. Luego de la lectura se le entregaron dos cuestionarios; uno cuestionario de dieciséis preguntas de opción múltiple para medir la comprensión y un segundo cuestionario de cuatro preguntas abiertas sobre la base de rediseño, solución de problemas, predicción y avisos conceptuales para medir la “transferencia de aprendizajes”. Para cada cuestionario se dio un tiempo de 10 minutos haciendo un total de 20 minutos de evaluación.

Como resultado de esta investigación, los participantes en el grupo de lectura electrónica no tuvieron una puntuación significativamente más alta en la comprensión lectora en comparación con los participantes en el grupo de lectura de texto tradicional. Los lectores de texto electrónico exhibieron puntajes significativamente más altos en comparación con los lectores de libros de texto tradicionales en las pruebas de “transferencia de aprendizajes”.

Gertner considera que estos resultados son un hallazgo favorable para los centros educativos que abogan por un mayor uso de los dispositivos electrónicos, ya que indica que la comprensión y transferencia de aprendizajes en la lectura electrónica es igual o mayor que la de un libro de texto tradicional.

2.6 Trabajo: “Aplicaciones móviles: Juegos que transforman la educación”

Zhang E. y Loeb L. (2013)⁽⁴⁴⁾ titularon su investigación “Aplicaciones móviles: Juegos que transforman la educación”. Como problemática a resolver afirman que las técnicas educativas actuales son muy ineficientes ya que la escuela se ha convertido en un ambiente de aprendizaje forzado que con frecuencia presenta el material educativo en un formato de conferencia pasiva y poco interesante.



Figura II-7. Impresión de pantalla 1 del software de Zhang E. y Loeb L.

Hacen hincapié en la importancia de tomar en cuenta las distintas formas de aprendizaje de las personas a saber visuales, auditivos y kinestésicos; estas formas de comunicación-interacción son un método fundamental por el que las personas organizan y consolidan su comprensión de un tema. Adicionalmente afirman que el reciente desarrollo de la tecnología para la plataforma de teléfonos inteligentes se presta muy bien a la educación y por ello deben aplicarse a la misma.

Zhang E. y Loeb plantean desarrollar un juego móvil educativo de rol 2D que utiliza los avances en la tecnología y la investigación en neurociencia para aumentar el aprendizaje, eficiencia y según lo denominan sus autores "pegajosidad". Diseñado para los sistemas operativos *iPhone* y *Android* inspirado en los monstruos de colección de *Pokemon*, franquicia de Nintendo. Los aspectos educativos se centran en el módulo de I de Matemáticas del examen estandarizado que se usa para admisiones a las universidades en los Estados Unidos, conocido como *SAT* en el cual es muy importante obtener una puntuación muy alta. Dicha prueba tiene cuatro secciones diferentes: aritmética, álgebra, geometría y análisis estadístico. La idea es que los usuarios de este juego aprendan poco a poco, de manera repetitiva los cuatro componentes de matemática.



Figura II-8. Impresión de pantalla 2 del software de Zhang E. y Loeb L.

En el juego se han mantenido los aspectos del juego original: la evolución del monstruo, la elección de su propio motor de arranque personalizado, un mundo enorme para explorar, variedad de monstruos, la capacidad de capturar y obtener monstruos enemigos, diferentes afiliaciones elementales entre monstruos y un sistema de combate fortaleza / debilidad (ver Figuras II-7, II-8 y II-9). Este juego fue desarrollado utilizando el *SDK* de Corona (herramienta para el desarrollo de juegos y aplicaciones para *iOS*, *Android*, *Amazon Kindle Fire* y *Burnes&Noble Nook*), que a su vez utiliza el lenguaje de scripting *Lua*. Corona permite incorporar transiciones de pantalla y movimiento además que cuenta con un sistema integrado para la animación de *sprites* lo que permite la creación, implementación y uso de expresiones de caracteres 2D simples y animaciones de ataque 2D.

Para almacenar información usaron archivos *JavaScript Object Notation* denominado *JSON*, el cual es un formato ligero para el intercambio de datos que no requiere el uso de *XML* y permite importar de forma rápida, fácil y eficiente en Corona.

En cuanto al diseño general del software, posee varios estados de pantalla diferentes: la pantalla de inicio, conquistar castillos (Conquista), explorando el mundo (Aventura), batallas y escenas de corte.



Zhang y Loeb diseñaron su juego pensando en tres públicos diferentes. La primera son los estudiantes que están motivados para estudiar SAT. La segunda está destinada a los padres que quieren que sus hijos aprendan sin obligarlos a memorizar libros y el último, los distritos escolares.

Entre sus conclusiones los autores recomiendan a los desarrolladores de aplicaciones móviles o *tablets* el movimiento multiplataforma, el cual consiste en la integración a través de los espacios. La idea es que un usuario pueda permanecer conectado o al alcance de una plataforma de software en cualquier lugar que se encuentre. Por ejemplo, que los usuarios puedan moverse sin problemas desde una plataforma de ordenador personal, teléfono, a un comprimido al llegar a la escuela.

CAPÍTULO III:

PROPUESTA DE TRABAJO ESPECIAL DE GRADO

A continuación se describe la propuesta de trabajo especial de grado, se describe el objetivo general y los objetivos específicos que se plantearon para gradualmente alcanzar tal fin. También se delimita el problema a tratar y las razones para abordarlo, así como también se establecen el alcance y los límites del proyecto.

Adicionalmente se detallan los requisitos, se desglosan las funcionalidades así como también sus características, tomando en cuenta tanto los requisitos funcionales como no funcionales pensados para la aplicación.

3.1 Planteamiento del problema

El problema global a resolver es, por una parte, la motivación de niños en edad escolar para aprender a leer, y por otra, la efectividad del método utilizado.

En particular, se observa lo siguiente:

- Generalmente, las estrategias de enseñanza de lectura aplicadas no consideran la diversidad en los modos de aprender de cada estudiante en particular, sino que se utiliza una metodología y una velocidad de avance uniforme para todo el grupo.

- Las aplicaciones existentes, debido a la complejidad en su interfaz, requieren la supervisión permanente de alguien que sepa leer para navegar por sus secciones o interactuar, lo que excluye a los niños que aún no saben leer a aprender por sí mismos.

- Muchas de esas aplicaciones no ofrecen opciones de entretenimiento como complemento al proceso de aprendizaje, por lo cual los niños se aburren al poco tiempo de uso.

Para resolver este problema, se propone una solución cuyos objetivos se detallan más adelante.

3.2 Justificación

Los dispositivos móviles se han convertido en un éxito en los últimos años. Sus componentes de hardware y software hacen posible el uso de nuevas formas de interacción, específicamente, pantallas táctiles y la interpretación de gestos. Esto permite que grupos de usuarios no especializados en el manejo de los componentes tradicionales (teclado, mouse, pantalla no táctil), particularmente, los niños pequeños, manejen con facilidad aplicaciones en estos dispositivos debido a que la interacción les resulta muy intuitiva.

Además, enseñar idiomas a niños en edad preescolar o en los primeros años de educación básica resulta difícil si no se sienten motivados, ya que el aprendizaje requiere mantener un nivel constante de interés y trabajo. Por lo tanto, una aplicación que los haga sentir cómodos y entretenidos, derribará esa potencial barrera y hará que el proceso de aprendizaje parezca un juego.

Por otro lado, el aprendizaje de vocabulario es un aspecto crítico en el estudio de cualquier idioma, ya que es necesario para aplicar los demás conocimientos y habilidades relacionadas (como el entendimiento del lenguaje hablado y la lectura). Por esto, el uso de juegos para presentar y ejercitar vocabulario se convierte en un factor importante. El uso de ilustraciones coloridas, sonidos, diagramación adecuada, animaciones, y otros elementos permite llamar la atención del alumno, teniendo en cuenta la forma en que aprende de acuerdo a sus diferencias individuales en relación a la interpretación del mundo.

Adicionalmente, las tendencias económicas y tecnológicas a nivel global apuntan a un continuo incremento en la cantidad de equipos tecnológicos con que cuentan los estudiantes. Por lo tanto, se hace necesario aprovechar estos beneficios creando materiales que puedan ser utilizados y signifiquen un aporte al proceso educativo.

Las soluciones tecnológicas que pueden ayudar a mejorar la motivación presentan complejidades en distintos niveles. Por un lado, la interpretación de gestos y la usabilidad de la aplicación a implementar son puntos críticos debido a la naturaleza de los usuarios, como lo muestran distintos estudios sobre la interacción entre niños de primaria y dispositivos táctiles. Por otro lado, la forma en que se guardan y despliegan los contenidos multimedia de naturaleza heterogénea constituye un desafío importante, ya que se deben integrar simultáneamente muchos elementos de distinto formato, sin afectar el rendimiento.

3.3 Objetivo general

Facilitar y motivar el proceso de aprender a leer en niños de tres hasta seis años de edad mediante el uso de una aplicación interactiva para dispositivos *Android*.

3.4 Objetivos específicos

- Elaborar el diseño lógico, funcional y físico del software educativo considerando el nivel de los usuarios, utilizando el Método de Enseñanza Mixto.
- Implementar las diferentes actividades del software educativo para su uso en la plataforma *Android*.
- Adaptar el software educativo para su uso en los diferentes dispositivos móviles *Android*.

- Hacer pruebas de las funcionalidades de los diferentes módulos y etapas de los prototipos desarrollados y hacer las correcciones necesarias.
- Liberar el producto final de software educativo para su descarga y uso.

3.5 Alcance

El alcance de este trabajo consiste en diseñar y desarrollar una aplicación móvil, para su uso en dispositivos *Android* versión 4.0 o superior, aprovechando el uso de la pantalla táctil, la corneta integrada y los controles por movimiento del dispositivo. La aplicación está diseñada para ayudar en el proceso de la enseñanza de la lectura de las cinco vocales y diecisiete consonantes del castellano.

3.6 Análisis del proyecto

- Restricciones: la aplicación está limitada por un conjunto de restricciones en distintos niveles, lo que afecta tanto el diseño como la implementación (restricciones en cuanto a la capacidad técnica de la tecnología del hardware, como lo son la potencia de procesamiento, cantidad de memoria disponible, etc; así como las limitaciones propias del sistema operativo *Android*).
- Texto: como la aplicación está dirigida a niños de entre 3 y 6 años, se asume que no saben leer o que recién están aprendiendo a hacerlo. Esta limitación afecta distintos aspectos del diseño, como la identificación de botones.
- Interacción: los niños de 3 a 6 años poseen un desarrollo psicomotor incompleto. Específicamente, destrezas como la motricidad fina no se encuentran completamente desarrolladas. Esto afecta el desarrollo de la aplicación al momento de diseñar e implementar los tamaños de los botones y la interacción táctil involucrada.
- Distribución espacial: debido a las características propias de la edad de los usuarios, estos tienden a concentrarse en un solo elemento a la vez. Esto

puede causar situaciones inesperadas al momento de interactuar con una aplicación de pantalla táctil. Por ejemplo, si el usuario toca inadvertidamente un botón con la mano que no está ocupando, y la pantalla cambia de vista, puede frustrarse al no identificar la causa del evento. Por esto, es necesario distribuir apropiadamente los elementos en la pantalla, de manera que tales errores se puedan evitar dentro de un escenario normal de uso.

- Características de los usuarios: la aplicación está dirigida a niños en edad escolar insertos en el sistema educativo, cuyas características son las siguientes:

Edad: 3 a 6 años.

Sexo: masculino y femenino.

Nivel educativo: Segundo y Tercer nivel de Educación Preescolar y Primer Grado de Educación Básica.

3.7 Requisitos funcionales (estrategia de la solución):

La aplicación introducirá a los niños en la lectura de las 5 vocales y las primeras 17 consonantes del castellano, prioridad basada en la dificultad y frecuencia de uso en el lenguaje, de cada consonante. Todas las vocales y cada una de las 17 consonantes están representadas en la aplicación en una clase respectiva.

3.7.1 Clases y actividades

La clase de vocales tiene a su vez dos actividades, organizadas de la siguiente manera:

- Una iteración de la actividad 1 “Toca cada vocal/sílaba dos veces”, en este caso se presentan cinco vocales, el niño deberá tocar dos veces cada una. Sobre estos botones se muestran ayudas visuales. Adicionalmente el sistema incorpora instrucciones verbales enlazadas a cada ayuda visual. El propósito de esta actividad es repasar el conocimiento del niño con respecto a cada vocal.

- Cinco iteraciones de la actividad 2 “Toca la vocal/sílaba”, se le requerirá al niño tocar la vocal requerida a través de una instrucción verbal y un mensaje de texto en pantalla. Esto deberá hacerlo con todas las vocales que se le presenten. En caso de que el niño tocara el botón incorrecto se le invitara por voz a intentarlo de nuevo. Adicionalmente el sistema incorpora ayudas auditivas enlazadas a cada ayuda visual. El propósito de esta actividad es que el niño pueda asociar e identificar su conocimiento con lo que se le está requiriendo.

Cada clase de consonantes tiene cuatro actividades diferentes, organizadas de la siguiente manera:

- Una iteración de la actividad 1 “Toca cada vocal/sílaba dos veces”, en este caso se presentan cinco sílabas y el niño deberá tocar dos veces cada una. Sobre estos botones se muestran ayudas visuales. Adicionalmente el sistema incorpora instrucciones verbales enlazadas a cada ayuda visual. El propósito de esta actividad es repasar el conocimiento del niño con respecto a cada sílaba.
- Cinco iteraciones de la actividad 2 “Toca la vocal/sílaba”, se le requerirá al niño tocar la sílaba requerida a través de una instrucción verbal y un mensaje de texto en pantalla. Esto deberá hacerlo con todas las sílabas. En caso de que el niño tocara el botón incorrecto se le invitara por voz a intentarlo de nuevo. Adicionalmente el sistema incorpora instrucciones verbales enlazadas a cada ayuda visual. El propósito de esta actividad es que el niño pueda asociar e identificar su conocimiento con lo que se le está requiriendo.
- Cuatro iteraciones de la actividad 3 “Toca cada palabra dos veces”, en la cual se presentaran tres palabras para que el niño toque dos veces cada una, y de esta manera escuchar cada palabra. Adicionalmente el sistema

incorpora instrucciones verbales enlazadas a cada ayuda visual. Con esta actividad se desea reforzar la memorización de palabras, su pronunciación y gramática.

- Cuatro iteraciones de la actividad 4 “Toca cada oración tres veces”, en la que se presenta una oración y un refuerzo visual. El niño deberá tocar tres veces la oración, y de esta manera oírla. Adicionalmente el sistema incorpora instrucciones verbales enlazadas a cada ayuda visual. Esta actividad es una etapa más compleja ya que se presenta la integración de las sílabas y palabras vistas en oraciones con significado.

La estructura de las clases está organizada de la siguiente manera:

Clase #:	Letra:	Actividad : Iteraciones
1	Vocales (a, e, i, o, u).	1:1 / 2:5
2	m	1:1 / 2:5 / 3:4 / 4:4
3	p	1:1 / 2:5 / 3:4 / 4:4
4	l	1:1 / 2:5 / 3:4 / 4:4
5	s	1:1 / 2:5 / 3:4 / 4:4
6	t	1:1 / 2:5 / 3:4 / 4:4
7	n	1:1 / 2:5 / 3:4 / 4:4
8	d	1:1 / 2:5 / 3:4 / 4:4
9	c*	1:1 / 2:3* / 3:4 / 4:4
10	c, q	1:1 / 2:5 / 3:4 / 4:4
11	b	1:1 / 2:5 / 3:4 / 4:4
12	v	1:1 / 2:5 / 3:4 / 4:4
13	rr	1:1 / 2:5 / 3:4 / 4:4
14	r	1:1 / 2:5 / 3:4 / 4:4
15	f	1:1 / 2:5 / 3:4 / 4:4
16	j	1:1 / 2:5 / 3:4 / 4:4
17	g	1:1 / 2:5 / 3:4 / 4:4
18	ñ	1:1 / 2:5 / 3:4 / 4:4

Tabla N°1. Estructura de las clases. Fuente: Carlos Carvajal 2015.

*En el caso de la letra “c” la actividad 2 solo se itera tres veces, ya que esa clase solo trata las sílabas “ca”, “co” y “cu”. Se hizo de esta manera para contrastar su pronunciación con las sílabas en la clase siguiente (clase “c,q”).

3.7.2 Mini-juegos

Con la finalidad de relajar la mente del niño y de agregarle diversión al proceso de aprendizaje, cada dos clases se añadirá uno de los siguientes mini-juegos:

- *Fruit Ninja*: en este juego el niño deberá utilizar su dedo como si fuera una espada para cortar las manzanas que aparecerán en pantalla. Por cada manzana cortada el niño recibirá un punto. El niño dispondrá de 60 segundos para lograr la mayor cantidad de puntos posibles.
- *Roll a Ball*: en este juego el niño utilizará los acelerómetros del dispositivo para hacer que una bola ruede a través de un plano con el objetivo de que toque todos los cubos con letras dispuestos sobre el mismo. El objetivo de este juego es que el niño recolecte todas las letras.
- *Circuit Racer*: en este juego el niño hará correr un vehículo en una pista circular y deberá completar las vueltas requeridas antes de agotarse el tiempo. Se disponen de tres vehículos con diseños y características de manejo diferentes, tres niveles de dificultad incremental y tres diseños de pista diferentes.
- *Angry Letters*: en este juego el niño hará uso de una honda mediante la cual arrojará bolas de madera sobre cubos pintados con letras. El objetivo es impactar con dichas bolas a todos los cubos de letras en tres intentos o menos.

Los mini-juegos están distribuidos a lo largo de la aplicación, de acuerdo a su dificultad, de la siguiente manera:

Ubicación (al completar la clase):	Nombre del juego:
2 (m)	<i>Fruit Ninja</i>
4 (l)	<i>Roll a Ball</i>
6 (t)	<i>Fruit Ninja</i>
8 (d)	<i>Roll a Ball</i>
10 (c,q)	<i>Circuit Racer</i>
12 (v)	<i>Fruit Ninja</i>
14 (r)	<i>Angry Letters</i>
16 (j)	<i>Circuit Racer</i>
18 (ñ)	<i>Angry Letters</i>

Tabla N°2. Distribución de los mini-juegos en la aplicación. Fuente: Carlos Carvajal 2015.

3.7.3 Menú principal

El menú principal de la aplicación le permite al usuario seleccionar entre cinco opciones, que son:

- “Aprende ya” (o “Continuar...”): bajo esta opción el usuario podrá iniciar el curso de aprender a leer (caso “Aprende ya”); o si ya tiene el curso iniciado, continuar en la clase siguiente a la última completada (caso “Continuar...”).
- “Salir del juego”: esta opción debe permitirle al usuario poder salir de manera segura de la aplicación al sistema operativo.
- “Seleccionar clase”: esta opción le permitirá al usuario escoger en un catálogo, la clase en la que desee participar de entre todas las que haya completado más la siguiente a estas.
- “Reiniciar app”: esta opción le brinda al usuario la oportunidad de borrar el progreso acumulado a lo largo del curso, colocándola en el mismo estado como viene de fábrica.

- “Créditos”: esta opción permite ver cierta información de las personas involucradas en el desarrollo de la aplicación.

3.8. Requisitos no funcionales

Son requisitos complementarios o atributos de calidad. Especifican criterios que juzgan operaciones del sistema en lugar de su comportamiento (requisitos funcionales).

3.8.1 Documentación

- La codificación del sistema debe ser clara y estar documentada de manera que algún programador pueda agregar funcionalidad posteriormente, procurando seguir los estándares de programación *Android*.

3.8.2 Seguridad

- Para evitar el borrado accidental de datos de progreso de la aplicación se implementa un menú de verificación por palabra clave.
- Se implementa la función de salida segura al sistema, para evitar la salida de la aplicación al presionar accidentalmente la tecla retroceder.

3.8.3 Actualización y portabilidad

- Disponibilidad para todo tipo de dispositivo *Android*, ya sean *smartphones* o *tablets*.
- La aplicación soporta la mayoría de las versiones de *Android* del mercado.
- Es necesario disponer de una conexión activa a internet, ya sea por *Wi-fi* o por plan de datos, para descargar futuras actualizaciones del producto.

3.8.4 Interfaz y usabilidad

- La aplicación consta de una interfaz sencilla, atractiva e intuitiva. De esta forma su uso no supone ningún impedimento o esfuerzo excesivo al usuario a la hora de disfrutar de la aplicación.
- La introducción de datos ha sido estructurada procurando evitar errores.
- La aplicación está diseñada para su uso en pantallas de alta definición (*Full HD*, 1080p) y a su vez escala automáticamente de manera óptima a resoluciones superiores e inferiores a esta.

3.8.5 Rendimiento

- Se esperan tiempos de respuesta no superiores a un segundo bajo cualquier petición.
- Se diseña la aplicación para que consuma muy pocos recursos de procesador y memoria, de manera que pueda ejecutarse en dispositivos antiguos o de baja potencia.
- Se tienen en consideración los dos puntos postulados anteriores manteniendo la interfaz a la mayor resolución y calidad de imagen y audio posible.

3.9 Metodología de desarrollo

Para el desarrollo de este trabajo de grado se utilizó el método de desarrollo *Extreme Programming*, debido a su orientación a producir código fuente funcional para grupos de desarrollo pequeños (*Extreme Programming* se explica detalladamente en el [apéndice B](#), al final de este documento).

Las fases que se siguieron en el desarrollo de esta aplicación se ilustran mediante el gráfico sencillo que sigue a continuación:

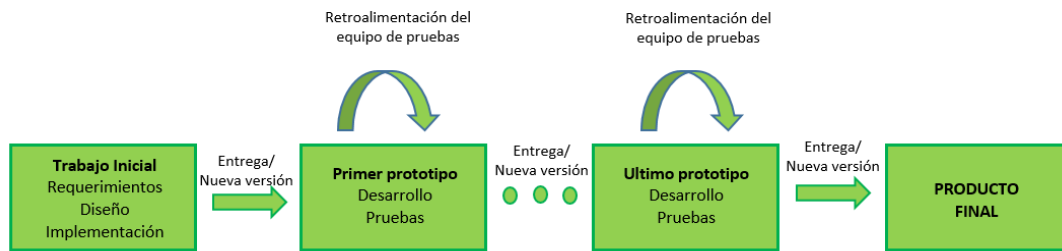


Figura III-1. Representación gráfica de la elaboración de prototipos. Carlos Carvajal, 2015.

3.10 Tecnologías a utilizar

Para llevar a cabo la estrategia de solución antes planteada, la cual permite alcanzar el objetivo general propuesto, se utilizan las siguientes tecnologías.

3.10.1 Lenguajes de programación

Unity soporta los lenguajes de programación *C#* y *Javascript*. Ambos lenguajes ofrecen el mismo desempeño ya que el *wrapper* de *Unity* intercambia el lenguaje a *Javascript* para las aplicaciones *Android*. Sin embargo, por preferencia del programador, se utilizará *C#* en la mayoría del código de la aplicación.

3.10.2 Almacenamiento de datos

Dado que la aplicación solo necesita almacenar el estado de progreso del usuario, se utiliza el método *PlayerPrefs* ("Last Level",int) de *Unity* para el almacenamiento y carga de la variable *int* (número entero) que guarda el número de la última clase iniciada por el usuario en el dispositivo.

La carga y almacenamiento se realiza mediante *PlayerPrefs.GetInt* y *PlayerPrefs.SetInt*, respectivamente.

De la misma manera, la función de reinicio de la app lo que hace es sobre-escribir el valor inicial, en dicha variable.

3.10.3 Interfaz

Para la interfaz se consideran dos aspectos importantes:

- El uso del método *OnGUI()* de *Unity* como una manera razonable de estandarizar por código todas las interfaces e interacciones de las clases. Esta opción reduce el nivel de personalización gráfica, pero permite una estandarización más precisa y de una manera más simple para el programador.
- El uso del método *GUI* de *Unity* como una manera de mejorar la apariencia en ciertas interacciones que requieren algo más de personalización gráfica. Esta opción permite una mayor personalización gráfica, pero requiere mucha mayor configuración y tiempo por parte del programador.

Dada la gran cantidad de clases y actividades, y los requerimientos específicos de funcionalidad y apariencia, se opta por utilizar ambos alcances en dos aspectos diferentes de cada actividad.

- El área de los botones, en la que es más importante la actividad y no se requiere mayor apariencia visual, será manejada por el método *OnGUI()*.
- Las áreas de refuerzo visual y de fondos de la aplicación, donde se requieren mayor calidad de imagen que funcionalidades, serán manejadas por el método *Unity GUI*.

Este diseño dual permite mejorar notablemente el desempeño gráfico de la aplicación ya que mucho del procesamiento se hace a través de código *OnGUI()*, que es mucho más simple que el que genera el editor de *Unity*.

3.10.4 Plataforma de desarrollo y pruebas

Para el desarrollo de este trabajo especial de grado se utilizan de las siguientes herramientas:

- **Android Developer Tools (ADT):** es un conjunto de herramientas y librerías para el desarrollo de software para la plataforma *Android*. Incluye un simulador de *Android* para *PC* con *Windows*.
- **Eclipse:** entorno de desarrollo libre que permite la creación de aplicaciones *Java* y *Android* (*Android* a través del *plug-in* para *Android Developer Tools*).
- **Unity:** es un software de desarrollo que incluye numerosas librerías que ayudan al desarrollo rápido y de manera visual de juegos y aplicaciones similares, para un gran número de plataformas entre las cuales se incluye *Android*.
- **Adobe Photoshop:** software para la edición de imágenes y *sprites*.
- **Blender:** software para la creación de modelos 3D.
- **Notepad++:** editor de texto con utilidades de programación
- **Suite Office 2013:** para elaborar y editar textos y presentaciones referentes a este trabajo.
- **Bluestacks:** emulador de *Android* para *Windows*.
- **Computador de implementación y pruebas:** computador con procesador *Intel Xeon X5650* a 3.8GHz, 12GB de *RAM* y sistema operativo *Windows 7 x64 SP1*, utilizado para instalar las herramientas y desarrollar el software.
- **Samsung Galaxy Note (GT-N7000):** teléfono inteligente *Android* 4.1 que será utilizado para realizar las pruebas de software.
- **Google Nexus 7 (versión 2012):** tableta *Android* 5.1 que será utilizada para realizar pruebas de software.

3.11 Planificación inicial del proyecto

Para el desarrollo de este proyecto se escoge el método ágil *XP*, debido a su gran flexibilidad y tolerancia a cambios.

A continuación se muestra la planificación de historias iniciales luego de reunirse por primera vez con el tutor de este trabajo especial de grado y recopilar los requerimientos funcionales del sistema.

Historia de Usuario	
Identificador: 01	Nombre: Grabación de voces
Usuario: Usuario	Prioridad: Alta
Programador: Carlos Carvajal	
Descripción: consiste en la grabación y edición de todas las voces y sonidos a utilizar en la aplicación.	

Historia de Usuario	
Identificador: 02	Nombre: Recopilación de imágenes
Usuario: Usuario	Prioridad: Alta
Programador: Carlos Carvajal	
Descripción: consiste en la creación, recopilación y edición de todas las imágenes y texturas a utilizar en la aplicación.	

Historia de Usuario	
Identificador: 03	Nombre: Toca cada vocal/sílaba 2 veces.
Usuario: Usuario	Prioridad: Alta
Programador: Carlos Carvajal	
Descripción: es la primera actividad de cada clase. El niño debe tocar al menos dos veces cada vocal o sílaba antes de pasar a la siguiente actividad. El propósito es que el niño haya oído varias veces el sonido asociado.	

Historia de Usuario	
Identificador: 04	Nombre: Toca la vocal/sílaba
Usuario: Usuario	Prioridad: Alta
Programador: Carlos Carvajal	
Descripción: es la segunda actividad de cada clase. El niño debe tocar la vocal o sílaba solicitada antes de pasar a la siguiente actividad. El propósito es que el niño analice y reconozca el sonido solicitado.	

Historia de Usuario	
Identificador: 05	Nombre: Toca cada palabra 2 veces
Usuario: Usuario	Prioridad: Alta
Programador: Carlos Carvajal	
Descripción: es la tercera actividad de cada clase. El niño debe tocar al menos dos veces cada palabra antes de pasar a la siguiente actividad. El propósito es que el niño pueda escuchar y ver en pantalla como las sílabas forman palabras de uso común.	

Historia de Usuario	
Identificador: 06	Nombre: Toca cada oración 3 veces
Usuario: Usuario	Prioridad: Alta
Programador: Carlos Carvajal	
Descripción: es la cuarta actividad de cada clase. El niño debe tocar al menos tres veces cada oración antes de pasar a la siguiente clase. El propósito es que el niño pueda escuchar y ver en pantalla como las sílabas forman palabras, y éstas a su vez forman oraciones de uso común.	

Historia de Usuario	
Identificador: 07	Nombre: Creación de clases
Usuario: Usuario	Prioridad: Alta
Programador: Carlos Carvajal	
Descripción: crear y organizar todas las carpetas y configuraciones de las clases en el editor de <i>Unity</i> .	

Historia de Usuario	
Identificador: 08	Nombre: Creación de actividades
Usuario: Usuario	Prioridad: Alta
Programador: Carlos Carvajal	
Descripción: crear, configurar y relacionar todas las instancias de las actividades, en sus respectivas instancias de clases, en el editor de <i>Unity</i> .	

Historia de Usuario	
Identificador: 09	Nombre: <i>Fruit Ninja</i>
Usuario: Usuario	Prioridad: Media
Programador: Carlos Carvajal	
Descripción: es un mini-juego donde el niño tiene 60 segundos para cortar las manzanas, utilizando su dedo como si fuese una espada ninja. Este juego se intercalará con las clases, es sencillo y su propósito es divertir y distraer al niño.	

Historia de Usuario	
Identificador: 10	Nombre: <i>Roll a Ball</i>
Usuario: Usuario	Prioridad: Media
Programador: Carlos Carvajal	
Descripción: es un mini-juego donde el niño debe hacer rodar la bola utilizando los acelerómetros del dispositivo (control de movimiento) y recolectar los cubos marcados con letras. Este juego se intercalará con las clases, es sencillo y su propósito es divertir y distraer al niño.	

Historia de Usuario	
Identificador: 11	Nombre: <i>Circuit Racer</i>
Usuario: Usuario	Prioridad: Media
Programador: Carlos Carvajal	
Descripción: es un mini-juego donde el niño debe hacer pasar el carro por la meta la cantidad de vueltas indicadas dentro de un tiempo específico, esto de acuerdo al nivel de	

dificultad elegido de entre los tres disponibles. Cada nivel de dificultad debe presenten un fondo de pista diferente. Se ofrecerán tres vehículos con diferentes aspectos de manejabilidad. El carro será controlado por una palanca virtual. Este juego se intercalará con las clases, es sencillo y su propósito es divertir y distraer al niño.

Historia de Usuario	
Identificador: 12	Nombre: <i>Angry Letters</i>
Usuario: Usuario	Prioridad: Media
Programador: Carlos Carvajal	
Descripción: es un mini-juego donde el niño debe utilizar su dedo para ajustar la fuerza y el ángulo de lanzamiento de una bola de madera en una honda, para impactarla contra las letras que están al otro lado de la pantalla. Se colocarán ciertos obstáculos para dificultar la tarea. El niño dispondrá de tres bolas de madera. Este juego se intercalará con las clases, es sencillo y su propósito es divertir y distraer al niño.	

Historia de Usuario	
Identificador: 13	Nombre: Menú Principal
Usuario: Usuario	Prioridad: Alta
Programador: Carlos Carvajal	
Descripción: el usuario ingresa al menú principal luego de pasar pantallas de Bienvenida. Allí debe seleccionar entre las 5 opciones disponibles (Aprende ya, Menú de clases, Salir, Créditos y Reiniciar app).	

Historia de Usuario	
Identificador: 14	Nombre: Aprende ya
Usuario: Usuario	Prioridad: Alta
Programador: Carlos Carvajal	
Descripción: el usuario empieza el curso, dirigiéndose automáticamente a la primera actividad de la primera clase.	

Historia de Usuario	
Identificador: 15	Nombre: Continuar
Usuario: Usuario	Prioridad: Alta
Programador: Carlos Carvajal	
Descripción: en el caso de que el usuario tenga al menos una clase terminada (y guardada), el texto del botón deberá cambiar a “Continuar...” y al presionarlo, debe continuar el curso en la primera actividad de la clase siguiente a la última completada.	

Historia de Usuario	
Identificador: 16	Nombre: Salir
Usuario: Usuario	Prioridad: Alta
Programador: Carlos Carvajal	
Descripción: el usuario utilizará este botón para salir de manera segura de la <i>app</i> .	

Historia de Usuario	
Identificador: 17	Nombre: Créditos
Usuario: Usuario	Prioridad: Baja
Programador: Carlos Carvajal	
Descripción: en esta opción el usuario podrá ver información acerca de las personas involucradas en la creación de esta aplicación.	

Historia de Usuario	
Identificador: 18	Nombre: Reiniciar app
Usuario: Usuario	Prioridad: Alta
Programador: Carlos Carvajal	
Descripción: con esta opción el usuario podrá reiniciar el avance del curso. Debe generarse un método que evite el borrado accidental del progreso, por parte del usuario.	

Historia de Usuario	
Identificador: 19	Nombre: Menú de clases
Usuario: Usuario	Prioridad: Alta
Programador: Carlos Carvajal	
Descripción: esta opción le permite al usuario iniciar la clase que desee, entre las que ha aprobado hasta el momento.	

Historia de Usuario	
Identificador: 20	Nombre: Integración de los módulos
Usuario: Usuario	Prioridad: Alta
Programador: Carlos Carvajal	
Descripción: se finaliza la creación de la aplicación integrando todos los módulos anteriores, es decir, clases, mini-juegos, menús, etc. en un solo paquete de aplicación.	

Historia de Usuario	
Identificador: 21	Nombre: Pruebas semifinales.
Usuario: Usuario	Prioridad: Alta
Programador: Carlos Carvajal	
Descripción: se dispone de pruebas semifinales de integración y de todas las funcionalidades de la aplicación como un solo conjunto.	

Historia de Usuario	
Identificador: 22	Nombre: Corrección de errores
Usuario: Usuario	Prioridad: Alta
Programador: Carlos Carvajal	
Descripción: se corrigen todos los “bugs” detectados en la aplicación. Esta etapa no solo evalúa funcionalidades, sino también las características estéticas de la integración.	

Historia de Usuario	
Identificador: 23	Nombre: Prueba final
Usuario: Usuario	Prioridad: Alta
Programador: Carlos Carvajal	
Descripción: se dispone la aplicación para ser probada nuevamente, antes de ser distribuida.	

Historia de Usuario	
Identificador: 24	Nombre: Distribución
Usuario: Usuario	Prioridad: Alta
Programador: Carlos Carvajal	
Descripción: se dispone la aplicación para ser distribuida en <i>App Stores</i> u otros medios.	

Junto con el levantamiento de las historias de usuario se procedió con el estudio y presentación de casos de uso, así como el plan de entrega del proyecto.

3.11.1 Casos de uso

A continuación se presentan los casos de uso estimados para este proyecto. Ver Figuras III-2, III-3 y III-4:

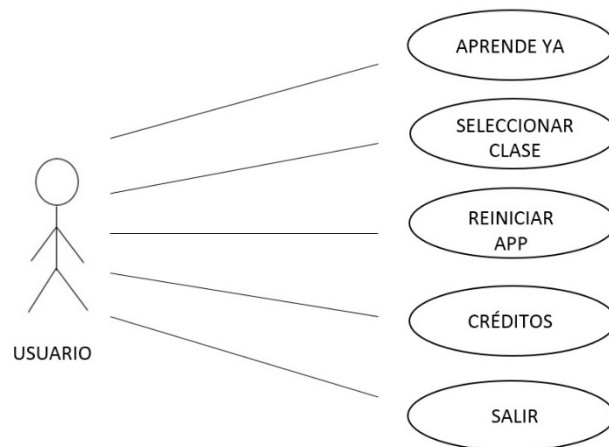


Figura III-2. Caso de Uso: Menú Principal. Fuente: Carlos Carvajal, 2015.

Caso de uso menú principal	
Precondiciones: el usuario debe haber iniciado la aplicación.	
Descripción: al usuario se le presentan una serie de situaciones a las que puede acceder.	
Condición de término: el usuario toca la opción salir, o hace doble toque al botón retroceder de su dispositivo <i>Android</i> .	
Condición de término fallida: se presiona el botón <i>home</i> del dispositivo <i>Android</i> .	
Actor: usuario.	
Flujo de eventos básico:	
Paso:	Acción:
1	Selecciona la opción "Aprende ya" o "Continuar..."
2	Si seleccionó la opción "Aprende ya", se carga la primera actividad de la primera clase. Si seleccionó la opción "Continuar...", se carga la primera actividad de la siguiente clase a la última que completó.
Flujo de eventos alternativo 1:	
Paso:	Acción:
1	El usuario selecciona la opción "Seleccionar clase".
2	Se cargan los menús de selección de clases.
Flujo de eventos alternativo 2:	
1	El usuario selecciona la opción "Reiniciar app".
2	Se carga la pantalla de "Reiniciar app".
Flujo de eventos alternativo 3:	
1	El usuario selecciona la opción "Créditos".
2	Se carga la pantalla de "Créditos".

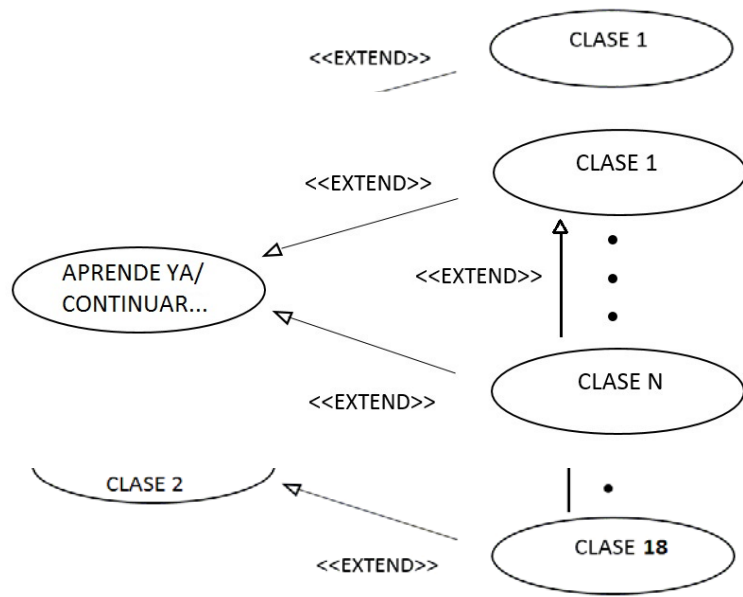


Figura III-4. Caso de Uso: Seleccionar clase. Fuente: Carlos Carvajal, 2015.

Caso de uso actividades
Precondiciones: - El usuario ha iniciado alguna clase del curso.

- El usuario ha completado alguna actividad o mini-juego de la clase.	
Descripción: se inicia la primera actividad de la clase, o se inicia otra actividad.	
Condición de término: el usuario completa la actividad.	
Condición de término fallida: se presiona el botón <i>home</i> del dispositivo <i>Android</i> .	
Actor: usuario.	
Flujo de eventos de inicio de curso:	
Paso:	Acción:
1	Se carga la primera actividad de la primera clase.
2	Si el usuario cumple el objetivo, se pasa a la siguiente actividad, y así sucesivamente.
Flujo de eventos de continuación del curso:	
1	Se carga la siguiente actividad de la clase, la primera actividad de la siguiente clase.
2	Si el usuario cumple el objetivo, se pasa a la siguiente actividad, clase o mini-juego, y así sucesivamente.
Flujo de eventos alternativo:	
1	Se hace doble toque del botón retroceder del dispositivo <i>Android</i> .
2	Se carga el menú principal.

Caso de uso reiniciar app	
Precondiciones: el usuario haya seleccionado la opción “Reiniciar app” en el menú principal.	
Descripción: el usuario desea reiniciar el progreso de la aplicación.	
Condición de término: se presiona el botón regresar.	
Condición de término fallida: se presiona el botón <i>home</i> del dispositivo <i>Android</i> .	
Actor: usuario.	
Flujo de eventos básico:	
Paso:	Acción:

1	Se muestra el procedimiento de reinicio.
2	El usuario escribe correctamente el texto en el campo.
3	Se presiona el botón “Reiniciar app”.
4	Se muestra el mensaje de éxito.
5	Se presiona el botón “Regresar” para retornar al menú principal.
Flujo de eventos alternativo:	
1	Se muestra el procedimiento de reinicio.
2	El usuario escribe incorrectamente el texto en el campo.
3	Se presiona el botón “Reiniciar app”.
4	Se muestra el mensaje de error.
5	Se presiona el botón “Regresar” para retornar al menú principal, o repite el procedimiento.

Caso de uso créditos	
Precondiciones: el usuario haya seleccionado la opción “Créditos” en el menú principal.	
Descripción: el usuario desea obtener información acerca de las personas involucradas con la creación de la aplicación.	
Condición de término: se presiona el botón regresar.	
Condición de término fallida: se presiona el botón <i>home</i> del dispositivo <i>Android</i> .	
Actor: usuario.	
Flujo de eventos básico:	
Paso:	Acción:
1	Se muestra la información de las personas involucradas con la creación de la aplicación.
2	Se presiona el botón “Regresar” para retornar al menú principal.

Caso de uso seleccionar clases	
Precondiciones: el usuario haya seleccionado la opción “Seleccionar clase” en el menú principal.	
Descripción: el usuario desea seleccionar una clase de aplicación en particular.	
Condición de término: se presiona el botón regresar, o se carga alguna clase.	
Condición de término fallida: se presiona el botón <i>home</i> del dispositivo <i>Android</i> .	
Actor: usuario.	
Flujo de eventos básico:	
Paso:	Acción:
1	Se muestran las clases disponibles en color y las clases no disponibles en colores opacos.
2	Si la clase no está en la lista, el usuario presiona el botón “Siguiente” para ver la siguiente lista de clases.
3	El usuario toca la clase que desea cargar.
4	Si está habilitada, se carga la primera actividad de la misma.
Flujo de eventos alternativo:	
1	Se muestran las clases disponibles en color y las clases no disponibles en colores opacos.
2	Si la clase no está en la lista, el usuario presiona el botón “Siguiente” para ver la siguiente lista de clases.
3	El usuario toca la clase que desea cargar.
4	Si la clase no está habilitada, se muestra un mensaje explicativo.

Flujo de eventos alternativo 2:	
1	Se presiona el botón “Regresar” o el botón retroceder en el dispositivo <i>Android</i> , para retornar al menú anterior de clases o al menú principal.

3.11.2 Plan de entrega

Se acuerda un tiempo de entrega de entre cuatro (4) y cinco (5) meses. En base a esto se procedió a la planificación de la primera iteración del proyecto.

3.12 Primera iteración

A continuación se lista la documentación de la planificación, diseño, codificación y pruebas que tuvieron lugar durante la primera iteración del proyecto.

3.12.1 Planificación

A continuación se lista la documentación de la planificación de la primera iteración

3.12.1.1 Historias de usuario

Historia de Usuario	
Identificador: 01	Nombre: Grabación de voces
Usuario: Usuario	Prioridad: Alta
Programador: Carlos Carvajal	

Descripción: consiste en la grabación y edición de todas las voces y sonidos a utilizar en la aplicación.

Historia de Usuario	
Identificador: 02	Nombre: Recopilación de imágenes
Usuario: Usuario	Prioridad: Alta
Programador: Carlos Carvajal	
Descripción: consiste en la creación, recopilación y edición de todas las imágenes y texturas a utilizar en la aplicación.	

Historia de Usuario	
Identificador: 03	Nombre: Toca cada vocal/sílaba 2 veces.
Usuario: Usuario	Prioridad: Alta
Programador: Carlos Carvajal	
Descripción: es la primera actividad de cada clase. El niño debe tocar al menos dos veces cada vocal o sílaba antes de pasar a la siguiente actividad. El propósito es que el niño haya	
Historia de Usuario	
Identificador: 04	Nombre: Toca la vocal/sílaba
Usuario: Usuario	Prioridad: Alta
Programador: Carlos Carvajal	
Descripción: es la segunda actividad de cada clase. El niño debe tocar la vocal o sílaba solicitada antes de pasar a la siguiente actividad. El propósito es que el niño analice y reconozca el sonido solicitado.	

Historia de Usuario	
Identificador: 05	Nombre: Toca cada palabra 2 veces
Usuario: Usuario	Prioridad: Alta

Programador: Carlos Carvajal
Descripción: es la tercera actividad de cada clase. El niño debe tocar al menos dos veces cada palabra antes de pasar a la siguiente actividad. El propósito es que el niño pueda escuchar y ver en pantalla como las sílabas forman palabras de uso común.

Historia de Usuario	
Identificador: 06	Nombre: Toca la oración 3 veces
Usuario: Usuario	Prioridad: Alta
Programador: Carlos Carvajal	
Descripción: es la cuarta actividad de cada clase. El niño debe tocar al menos tres veces cada oración antes de pasar a la siguiente clase. El propósito es que el niño pueda escuchar y ver en pantalla como las sílabas forman palabras, y éstas a su vez forman oraciones de uso común.	

3.12.1.2 Plan de entrega

Luego de reunirse todos los actores del proyecto, se determina que para la primera entrega se desarrollarían las funcionalidades genéricas de las primeras seis (06) historias de usuario, ya que son el núcleo de la aplicación.

3.12.1.3 Velocidad del proyecto

Se estima un tiempo de desarrollo de seis (06) semanas para la realización de las historias de usuario, desde la historia número 1 hasta la número 6.

3.12.1.4 Tareas específicas

Tarea 1	
Identificador: 01	Número de historia: 1

Nombre: Grabar y editar todas las voces y sonidos.	Tipo: Alta
Inicio: 23/03/2015	Fin: 27/03/2015
Responsable: Carlos Carvajal	
Descripción: grabar y editar todas las voces y sonidos a ser utilizados en la aplicación. Se utilizará el formato <i>mp3</i> bajo los siguientes parámetros: tasa de <i>bits</i> variable, resolución de 16 <i>bits</i> , un canal (mono) y frecuencia de muestreo de 22050Hz.	
Pruebas de aceptación: - Verificar que todos los archivos de audio sean adecuados.	

Tarea 2	
Identificador: 02	Número de historia: 2
Nombre: Recopilar todas las imágenes y texturas.	Tipo: Alta
Inicio: 30/03/2015	Fin: 03/04/2015
Responsable: Carlos Carvajal	
Descripción: recopilar y editar todas las imágenes y texturas a ser utilizadas en la aplicación. Se utilizarán los formatos de imagen <i>PNG</i> y <i>JPG</i> bajo los siguientes parámetros: la calidad de compresión <i>JPG</i> debe ser 7 para mantener un tamaño y calidad balanceado, el lado mayor de la imagen debe reducirse a 1024 píxeles y mantener la tasa de aspecto. Las imágenes deben centrarse en el elemento a mostrar para no distraer la atención del niño. Las imágenes no deben poseer derechos de autor.	
Pruebas de aceptación: - Verificar que cada imagen y textura sea adecuada.	

Tarea 3	
Identificador: 03	Número de historia: 3
Nombre: Toca cada vocal/sílaba dos veces	Tipo: Alta
Inicio: 06/04/2015	Fin: 10/04/2015
Programador: Carlos Carvajal	

Descripción: es la primera actividad de cada clase. El niño debe tocar al menos dos veces cada oración antes de pasar a la siguiente clase. El propósito es que el niño haya oído varias veces el sonido asociado a la vocal o sílaba mostrada en pantalla.

Pruebas de aceptación:

- Verificar que al tocar cada vocal/sílaba se reproduzca el sonido respectivo de manera adecuada.
- Verificar que sea necesario tocar (y escuchar) cada sílaba al menos dos veces para continuar.
- Verificar que se impida la reproducción múltiple de dos o más botones al mismo tiempo.
- Verificar que se muestra el refuerzo visual con imágenes que utilicen la vocal o sílaba en curso, de manera adecuada.
- Verificar que se ofrece respuesta auditiva a las imágenes de refuerzo.
- Verificar que se impida la salida cuando se presiona accidentalmente la tecla de retroceso.
- Verificar que se muestren instrucciones auditivas y de texto al iniciar la actividad.

Tarea 4	
Identificador: 04	Número de historia: 4
Nombre: Toca la vocal/sílaba.	Tipo: Alta
Inicio: 13/04/2015	Fin: 17/04/2015
Programador: Carlos Carvajal	
Descripción: es la segunda actividad de cada clase. El niño debe tocar la vocal o sílaba requerida para poder pasar a la siguiente clase. El propósito es que el niño analice y reconozca el sonido solicitado.	
Pruebas de aceptación: - Verificar que al tocar cada vocal/sílaba se reproduzca el sonido respectivo de manera adecuada. - Verificar que sea necesario tocar (y escuchar) la sílaba indicada para continuar. - Verificar que en caso de toque incorrecto, se repita la instrucción nuevamente. - Verificar que se impida la reproducción múltiple de dos o más botones al mismo tiempo.	

- Verificar que se muestra el refuerzo visual con imágenes que utilicen la vocal o sílaba en curso, de manera adecuada.
- Verificar que se ofrece respuesta auditiva a las imágenes de refuerzo.
- Verificar que se impida la salida cuando se presiona accidentalmente la tecla de retroceso.
- Verificar que se muestren instrucciones auditivas y de texto al iniciar la actividad.

Tarea 5	
Identificador: 05	Número de historia: 5
Nombre: Toca cada palabra dos veces.	Tipo: Alta
Inicio: 20/04/2015	Fin: 24/04/2015
Programador: Carlos Carvajal	
Descripción: es la tercera actividad de cada clase. El niño debe tocar al menos dos veces cada palabra antes de pasar a la siguiente actividad. El propósito es que el niño pueda escuchar y ver en pantalla como las sílabas forman palabras de uso común.	
Pruebas de aceptación: - Verificar que al tocar cada palabra se reproduzca el sonido respectivo de manera adecuada. - Verificar que sea necesario tocar (y escuchar) cada palabra al menos dos veces. - Verificar que se impida la reproducción múltiple de dos o más botones al mismo tiempo. - Verificar que se muestra el refuerzo visual con imágenes que utilicen la palabra en curso, de manera adecuada. - Verificar que se ofrece respuesta auditiva a las imágenes de refuerzo. - Verificar que se impida la salida cuando se presiona accidentalmente la tecla de retroceso. - Verificar que se muestren instrucciones auditivas y de texto al iniciar la actividad.	

Tarea 6	
Identificador: 06	Número de historia: 6
Nombre: Toca cada oración tres veces.	Tipo: Alta

Inicio: 27/04/2015	Fin: 01/05/2015
Programador: Carlos Carvajal	
Descripción: es la cuarta actividad de cada clase. El niño debe tocar al menos tres veces cada oración antes de pasar a la siguiente clase. El propósito es que el niño pueda escuchar y ver en pantalla como las sílabas forman palabras, y éstas a su vez forman oraciones de uso común.	
Pruebas de aceptación: - Verificar que al tocar cada oración se reproduzca el sonido respectivo de manera adecuada. - Verificar que sea necesario tocar (y escuchar) cada palabra al menos tres veces. - Verificar que se impida la reproducción múltiple de dos o más botones al mismo tiempo. - Verificar que se muestra el refuerzo visual con imágenes que utilicen la palabra en curso, de manera adecuada. - Verificar que se ofrece respuesta auditiva a las imágenes de refuerzo. - Verificar que se impida la salida cuando se presiona accidentalmente la tecla de retroceso. - Verificar que se muestren instrucciones auditivas y de texto al iniciar la actividad.	

3.12.1.5 Reuniones

Durante esta primera iteración se llevaron a cabo reuniones cortas y de manera regular, en las que se discutían las tareas a desarrollar, problemas que se podrían presentar, o que se hayan presentado, y sus posibles soluciones. La duración de estas reuniones estuvo comprendida entre quince (15) y treinta (30) minutos.

3.12.2 Diseño

Una vez grabadas todas las voces, se procedió a la edición con el programa *Adobe Audition CC*. El formato de los archivos de voz es *mp3*, mono, 16 *bits*, 22050Hz.

Las imágenes fueron retocadas con el uso de *Adobe Photoshop CS6* y las imágenes se almacenaron en formato *JPG* o *PNG*.

Para organizar el trabajo, se crearon las respectivas carpetas en el editor de *Unity* y se importaron los archivos, quedando estructuradas las carpetas como se ven a continuación en la figura III-5:

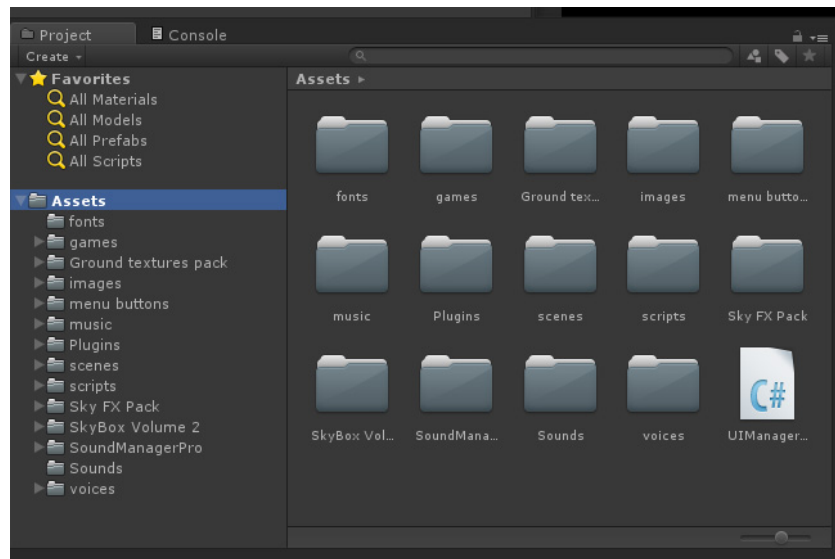


Figura III-5. Impresión de pantalla de la organización del proyecto en el editor de *Unity*.

El diagrama del sistema queda de la siguiente manera como se muestra en la figura III-6 a continuación:

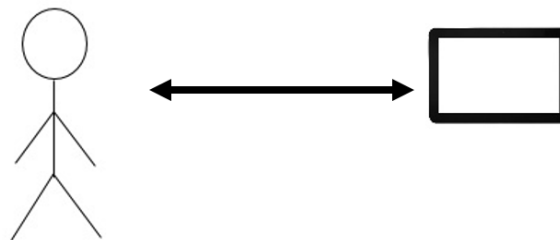


Figura III-6. Diagrama del sistema. Fuente: Carlos Carvajal 2015.

3.12.2.1 Diagrama de clases (*GameObjects*)

Cada clase (o, *script* o archivo de código) está enlazada a un *GameObject* (objeto cualquiera dentro de una escena de *Unity*), en cada escena en *Unity*. Cada actividad de la aplicación es en sí misma una escena en *Unity*.

El diagrama presentado como figura III-7 muestra de manera general y genérica la relación de clases, aplicable a cada una de las actividades creadas. (Ver también las figuras [A-1](#), [A-2](#), [A-3](#), [A-4](#), [A-5](#), [A-6](#), [A-7](#) en el [Apéndice A](#))

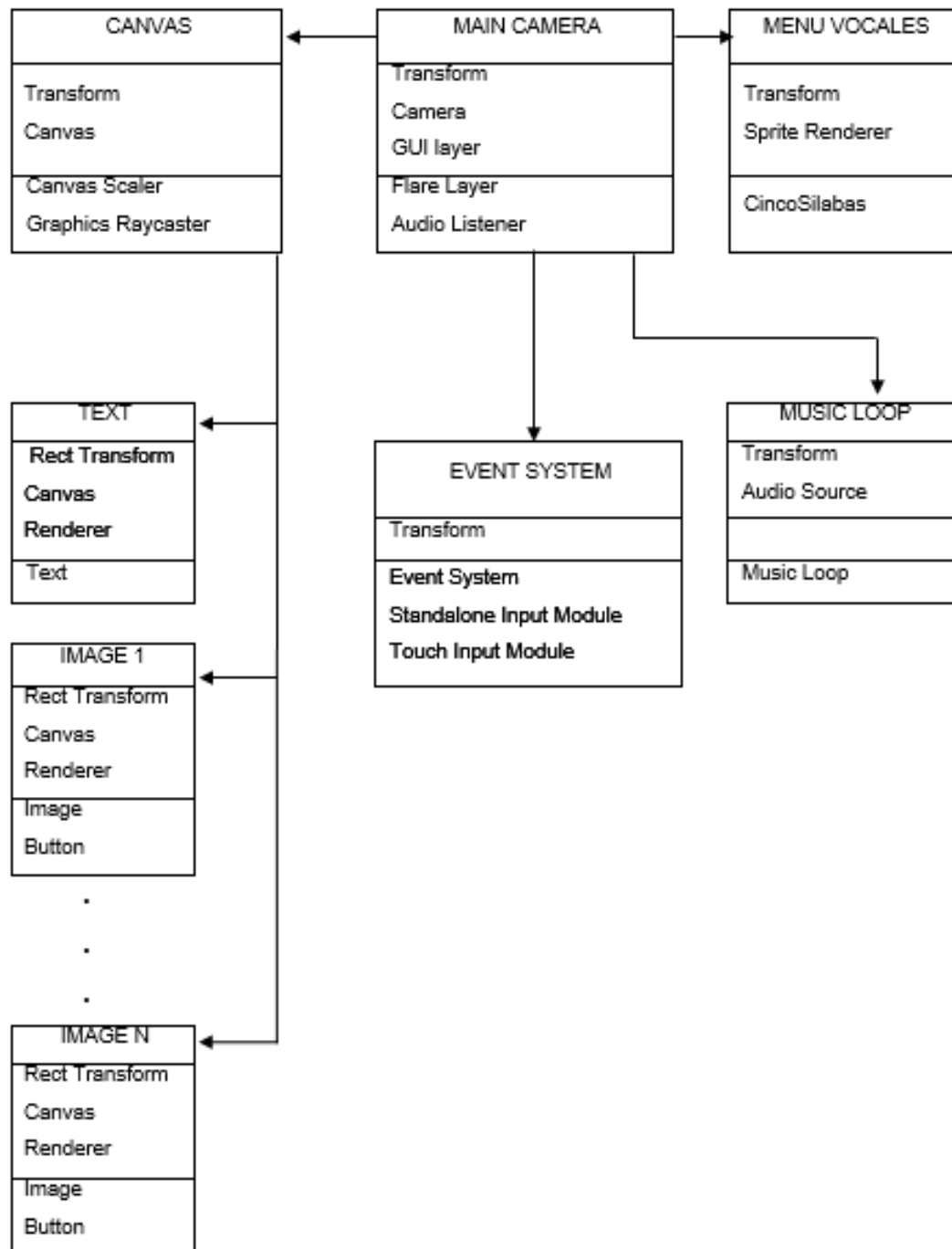


Figura III-7. Diagrama de *GameObjects* para las actividades. Fuente: Carlos Carvajal 2015.

3.12.3 Codificación

Primero fue necesario dividir la tarea en varias fases:

- Mostrar por pantalla los elementos necesarios para la interacción. Para tal fin se utilizaron dos métodos: la función *OnGUI* para la programación de los botones, y el uso de la interfaz visual *Unity GUI* para las ayudas visuales. Esto se hace para automatizar el trabajo de programación y garantizar la homogeneidad en las interfaces entre una actividad y otra.
- Cargar los archivos de audio y las imágenes, y relacionar a cada uno entre sí. Para facilitar esta tarea se utilizó el atributo público de ciertas variables de *Unity*, lo que permite cargar los archivos con el editor, de manera visual y más rápida.
- Para impedir la reproducción superpuesta del audio de los botones y para agregar una barra de progreso que ayudara a seguir la pronunciación de las sílabas, se crearon “enumeradores” y “co-rutinas” a las cuales se les asignan retrasos de tiempo fijos. La barra de progreso consiste en una textura (de partículas en este caso), que se va trasladando en el eje x, en una serie de iteraciones por un tiempo dado hasta llegar al final. Si el tamaño de la textura excede a la del botón o quedase a una fracción del mismo (cosa que ocurriría en ciertos tamaños de pantalla y por la imprecisión matemática de dicha relación de iteraciones-tiempo), la misma regresa a su posición inicial (no visible).

Entonces, se creó un *script* para manejar cada actividad en particular, y dado que se desea mantener el mismo fondo musical a lo largo de la clase (sin interrupciones, ni reinicio), se creó también el *script MusicLoop.cs* para tal fin.

3.12.4 Pruebas

Durante esta etapa se cargaron los archivos de audio de algunas clases y se realizaron una serie de pruebas utilizando el editor de *Unity*, basadas en las pruebas

de aceptación especificadas en las tareas de desarrollo. A continuación se puede observar en las figuras III-8 hasta la figura III-13 la aprobación de todas las pruebas (de un total de seis [6] pruebas) al finalizar esta primera iteración:

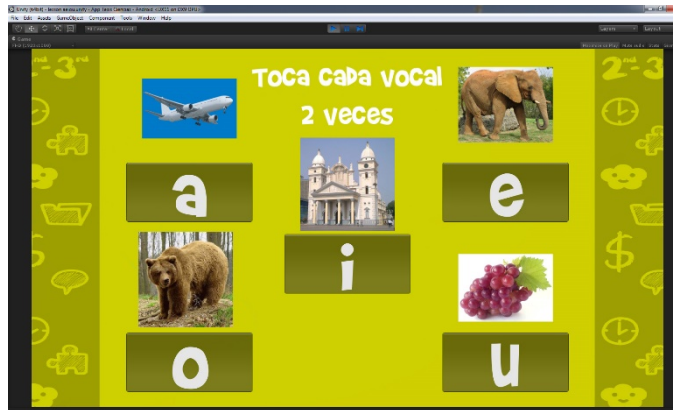


Figura III-8. Impresión de pantalla de la prueba de la actividad 1 “Toca cada vocal/sílaba dos veces”.

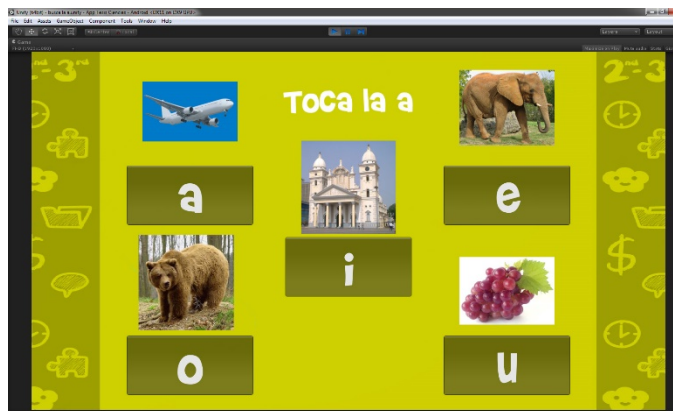


Figura III-9. Impresión de pantalla de la prueba de la actividad 2 “Toca cada la vocal/sílaba”.



Figura III-10. Impresión de pantalla de la prueba de la actividad 3 “Toca cada palabra dos veces”.



Figura III-11. Impresión de pantalla de la prueba de la actividad 4 “Toca cada oración tres veces”.



Figura III-12. Impresión de pantalla de la prueba de la actividad 1 “Toca cada vocal/sílaba dos veces” (para el caso de la clase “c”).



Figura III-13. Impresión de pantalla de la prueba de la actividad 2 “Toca la vocal/sílaba” (para el caso de la clase “c”).

3.13 Segunda iteración

Durante la segunda iteración se crearon todas las instancias de actividades y todas las instancias de las clases de la aplicación.

3.13.1 Planificación

A continuación se lista la documentación de la planificación de la segunda iteración.

3.13.1.1 Historias de usuario

Historia de Usuario	
Identificador: 07	Nombre: Creación de clases
Usuario: Usuario	Prioridad: Alta
Programador: Carlos Carvajal	
Descripción: crear y organizar todas las carpetas y configuraciones de las clases en el editor de <i>Unity</i> .	

Historia de Usuario	
Identificador: 08	Nombre: Creación de actividades
Usuario: Usuario	Prioridad: Alta
Programador: Carlos Carvajal	
Descripción: crear, configurar y relacionar todas las instancias de las actividades, en sus respectivas instancias de clases, en el editor de <i>Unity</i> .	

3.13.1.2. Plan de entrega

Para esta segunda iteración se acordó el desarrollo de las historias de usuario números 7 y 8, así como realizar una segunda entrega de un prototipo funcional de las historias de usuario antes mencionadas.

3.13.1.3. Velocidad del proyecto

Se acordó que esta iteración tendría un tiempo de entrega estimado de tres (03) semanas.

3.13.1.4. Tareas específicas

A continuación se muestra el conjunto de tareas de desarrollo.

Tarea 7	
Identificador: 07	Número de historia: 7
Nombre: Creación de clases.	Tipo: Alta
Inicio: 04/05/2015	Fin: 08/05/2015
Responsable: Carlos Carvajal	
Descripción: crear y organizar todas las carpetas y configuraciones de las clases en el editor de <i>Unity</i> .	
Pruebas de aceptación: - Verificar que todas las carpetas y la estructura de las clases estén cargadas y debidamente configuradas en el editor de <i>Unity</i>.	

Tarea 8	
Identificador: 08	Número de historia: 8
Nombre: Creación de actividades.	Tipo: Alta
Inicio: 11/05/2015	Fin: 22/05/2015
Responsable: Carlos Carvajal	
Descripción: crear, configurar y relacionar todas las instancias de las actividades, en sus respectivas instancias de clases, en el editor de <i>Unity</i> .	
Pruebas de aceptación: - Verificar la creación ordenada de cada clase. - Verificar la creación ordenada de cada actividad dentro de cada clase.	

- Verificar que cada actividad y que cada clase funcione de manera adecuada, y en el orden adecuado.
- Verificar que se haya cumplido el esquema de clases y actividades acordado.

3.13.1.5 Reuniones

Durante esta segunda iteración se llevaron a cabo reuniones cortas y de manera regular, en las que se discutían las tareas a desarrollar, problemas que se podrían presentar, o que se hayan presentado, y sus posibles soluciones. La duración de estas reuniones estuvo comprendida entre quince (15) y treinta (30) minutos.

3.13.2 Diseño

Se procede a crear las instancias de cada actividad en cada clase respectiva. Se mantiene el mismo diseño de la iteración anterior.

3.13.2.1. Diagrama de clases

Se procede a crear las instancias de cada actividad en cada clase respectiva. Se mantiene el mismo diagrama de clases de la iteración anterior.

3.13.3 Codificación

Se procede a crear las instancias de cada actividad en cada clase respectiva. Los códigos creados en la instancia anterior fueron diseñados para automatizar gran parte de este proceso. La asignación de variables públicas se hace a través del editor de *Unity*, tal como se ilustra en la imagen III-14 presentada a continuación:

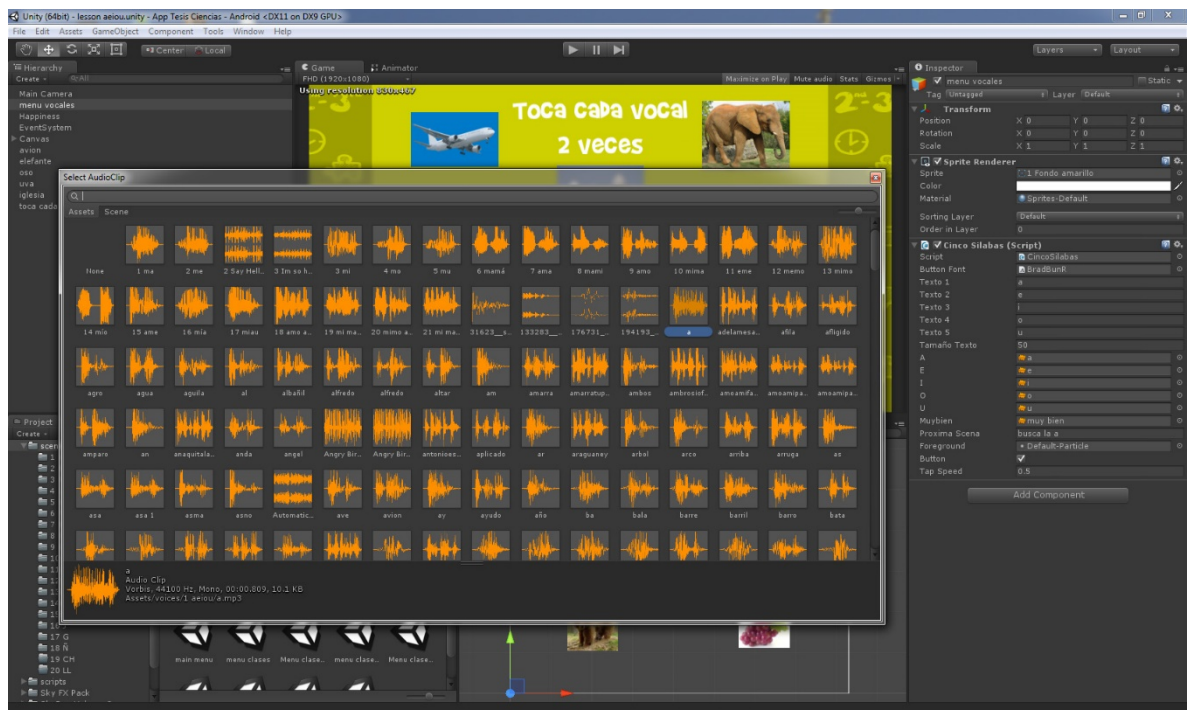


Figura III-14. Impresión de pantalla de carga de variables multimedia en el editor de Unity.

3.13.4 Pruebas

Durante esta etapa se cargaron todas las actividades en cada clase y se configuró cada una específicamente. Cabe destacar que cada actividad lleva su propio conjunto de imágenes y sonidos, por lo cual esta fase del proyecto lleva mucha atención y tiempo ya que, en el caso de las imágenes, cada una tiene dimensiones y aspectos diferentes y se hace necesario ajustar cada una de manera individual para mantener un alto nivel de homogeneidad en la apariencia de todas las clases.

Dado que la verificación de esta prueba representó la inspección de más de 200 escenas/actividades repartidas en las 18 clases, la prueba se realizó directamente en el prototipo instalado en la tableta *Android* de pruebas. No resulta conveniente hacer impresiones de tantas imágenes en este documento. Se presentan dos (02) ejemplos enumeradas como figura III-15 y figura III-16:

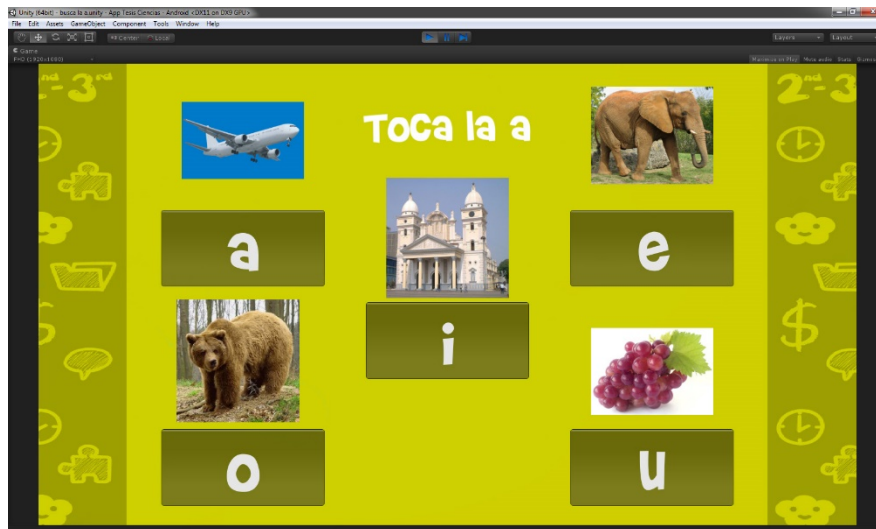


Figura III-15. Impresión de pantalla de una actividad con todos sus elementos cargados, en el simulador de *Unity*.



Figura III-16. Impresión de pantalla de otra actividad con todos sus elementos cargados, en el simulador de *Unity*.

3.14 Tercera iteración

Durante la tercera iteración se crearon todas las instancias de los mini-juegos de la aplicación.

3.14.1 Planificación

A continuación se lista la documentación de la planificación de la tercera iteración.

3.14.1.1 Historias de usuario

Historia de Usuario	
Identificador: 09	Nombre: <i>Fruit Ninja</i>
Usuario: Usuario	Prioridad: Media
Programador: Carlos Carvajal	
Descripción: es un mini-juego donde el niño tiene 60 segundos para cortar las manzanas, utilizando su dedo como si fuese una espada ninja. Este juego se intercalará entre las clases, es sencillo y su propósito es divertir y distraer al niño.	

Historia de Usuario	
Identificador: 10	Nombre: <i>Roll a Ball</i>
Usuario: Usuario	Prioridad: Media
Programador: Carlos Carvajal	
Descripción: es un mini-juego donde el niño debe hacer rodar la bola utilizando los acelerómetros del dispositivo (control de movimiento) y recolectar los cubos marcados con letras. Este juego se intercalará entre las clases, es sencillo y su propósito es divertir y distraer al niño.	

Historia de Usuario	
Identificador: 11	Nombre: <i>Circuit Racer</i>
Usuario: Usuario	Prioridad: Media
Programador: Carlos Carvajal	
Descripción: es un mini-juego donde el niño debe hacer pasar el carro por la meta la cantidad de vueltas indicadas dentro del tiempo indicado, esto de acuerdo al nivel de dificultad elegido de entre los tres disponibles. Cada nivel de dificultad debe presentar un fondo de pista diferente. Se ofrecerán tres vehículos con diferentes aspectos de	

manejabilidad. El carro será controlado por una palanca virtual. Este juego se intercalará entre las clases, es sencillo y su propósito es divertir y distraer al niño.

Historia de Usuario	
Identificador: 12	Nombre: <i>Angry Letters</i>
Usuario: Usuario	Prioridad: Media
Programador: Carlos Carvajal	
Descripción: es un mini-juego donde el niño debe utilizar su dedo para ajustar la fuerza y el ángulo de lanzamiento de una bola de madera en una honda, para impactarla contra las letras que están al otro lado de la pantalla. Se colocarán ciertos obstáculos para dificultar la tarea. El niño dispondrá de tres bolas de madera. Este juego se intercalará entre las clases, es sencillo y su propósito es divertir y distraer al niño.	

3.14.1.2. Plan de entrega

Para esta tercera iteración se acordó el desarrollo de las historias de usuario números 09, 10, 11 y 12, así como realizar una tercera entrega de un prototipo funcional de las historias de usuario antes mencionadas.

3.14.1.3. Velocidad del proyecto

Se acordó que esta iteración tendría un tiempo de entrega estimado de seis (06) semanas.

3.14.1.4. Tareas específicas

A continuación se muestra el conjunto de tareas de desarrollo.

Tarea 9	
Identificador: 09	Número de historia: 9
Nombre: <i>Fruit Ninja.</i>	Tipo: Alta
Inicio: 25/05/2015	Fin: 29/06/2015
Responsable: Carlos Carvajal	
Descripción: es un mini-juego donde el niño tiene 60 segundos para cortar las manzanas, utilizando su dedo como si fuese una espada ninja. Este juego se intercalará entre las clases, es sencillo y su propósito es divertir y distraer al niño.	
Pruebas de aceptación: - Verificar que el juego termine a los 60 segundos. - Verificar que las manzanas se generen en grupos de 4, en posiciones aleatorias, cada 2 segundos. - Verificar que las acciones de toque interactúen adecuadamente con el área de juego. - Verificar que la interfaz muestre la puntuación, el tiempo, el fin de juego y todo el escenario de juego de manera adecuada. - Verificar que los sonidos y la música funcionen adecuadamente. - Verificar el inicio y fin adecuados del juego.	

Tarea 10	
Identificador: 10	Número de historia: 10
Nombre: <i>Roll a Ball.</i>	Tipo: Alta
Inicio: 01/06/2015	Fin: 05/06/2015
Responsable: Carlos Carvajal	
Descripción: es un mini-juego donde el niño debe hacer rodar la bola utilizando los acelerómetros del dispositivo (control de movimiento) y recolectar los cubos marcados con letras. Este juego se intercalará entre las clases, es sencillo y su propósito es divertir y distraer al niño.	
Pruebas de aceptación: - Verificar que el juego termine al haber tocado todos los cubos de letras. - Verificar que los elementos del juego se generen y destruyan correctamente.	

- Verificar que las acciones de movimiento interactúen adecuadamente con el área de juego.
- Verificar que la interfaz muestre la puntuación, el fin de juego y todo el escenario de juego de manera adecuada.
- Verificar que los sonidos y la música funcionen adecuadamente.
- Verificar el inicio y fin adecuados del juego.

Tarea 11	
Identificador: 11	Número de historia: 11
Nombre: <i>Circuit Racer.</i>	Tipo: Alta
Inicio: 08/06/2015	Fin: 19/06/2015
Responsable: Carlos Carvajal	
Descripción: es un mini-juego donde el niño debe hacer pasar el carro por la meta la cantidad de vueltas indicadas dentro del tiempo indicado, esto de acuerdo al nivel de dificultad elegido de entre los tres disponibles. Cada nivel de dificultad debe presentar un fondo de pista diferente. Se ofrecerán tres vehículos con diferentes aspectos de manejabilidad. El carro será controlado por una palanca virtual. Este juego se intercalará entre las clases, es sencillo y su propósito es divertir y distraer al niño.	
Pruebas de aceptación: - Verificar que los menús del juego funcionen y registren las selecciones del usuario, todo esto de manera adecuada. - Verificar las condiciones de todos los estados de juego. - Verificar que los elementos del juego se generen y destruyan correctamente. - Verificar que las acciones de movimiento por el <i>stick</i> virtual interactúen adecuadamente con el carro. - Verificar que la interfaz muestre el estado del juego y todo el escenario de juego de manera adecuada. - Verificar que los sonidos y la música funcionen adecuadamente. - Verificar el inicio y fin adecuados del juego.	

Tarea 12	
Identificador: 12	Número de historia: 12
Nombre: <i>Angry Letters.</i>	Tipo: Alta
Inicio: 22/06/2015	Fin: 03/07/2015
Responsable: Carlos Carvajal	
Descripción: es un mini-juego donde el niño debe utilizar su dedo para ajustar la fuerza y el ángulo de lanzamiento de una bola de madera en una honda, para impactarla contra las letras que están al otro lado de la pantalla. Se colocarán ciertos obstáculos para dificultar la tarea. El niño dispondrá de tres bolas de madera. Este juego se intercalará entre las clases, es sencillo y su propósito es divertir y distraer al niño.	
Pruebas de aceptación: - Verificar las condiciones de todos los estados de juego. - Verificar que los elementos del juego se generen y destruyan correctamente. - Verificar que las acciones de movimiento por toque interactúen adecuadamente con el juego. - Verificar que se muestre el estado del juego y todo el escenario de juego de manera adecuada. - Verificar que los sonidos y la música funcionen adecuadamente. - Verificar el inicio y fin adecuados del juego.	

3.14.1.5 Reuniones

Durante esta segunda iteración se llevaron a cabo reuniones cortas y de manera regular, en las que se discutían las tareas a desarrollar, problemas que se podrían presentar, o que se hayan presentado, y sus posibles soluciones. La duración de estas reuniones estuvo comprendida entre quince (15) y treinta (30) minutos.

3.14.2 Diseño

Cada mini-juego tiene su propio conjunto de clases y scripts, para llevar a cabo sus propias tareas específicas.

Dada la complejidad intrínseca de cada mini-juego, así como el estado complementario de los mismos con respecto al propósito educativo de las clases de la aplicación, se optó por no almacenar registros, puntuación o estado de ninguno de los juegos.

3.14.2.1 *Fruit Ninja*

El diseño de este mini-juego fue basado en otros juegos similares existentes. Se elige este tipo de juego ya que es lo suficientemente sencillo, fácil de usar y entender por cualquier niño a los que va dirigida esta aplicación.

El juego consiste en cortar la mayor cantidad de manzanas con el dedo, mientras duren los 60 segundos dispuestos para ello. El niño debe tocar la pantalla con su dedo simulando ser una espada. El juego es totalmente automático. Las manzanas son lanzadas de manera automática y mientras suben y bajan, deben ser cortadas por el niño.

Diagrama de clases (*GameObjects*)

A continuación se presenta la figura III- 17 con el diagrama de clase del mini-juego *Fruit Ninja*. (Ver también las figuras [A-8](#), [A-9](#), [A-10](#), [A-11](#) y [A-12](#) en el [Apéndice A](#))

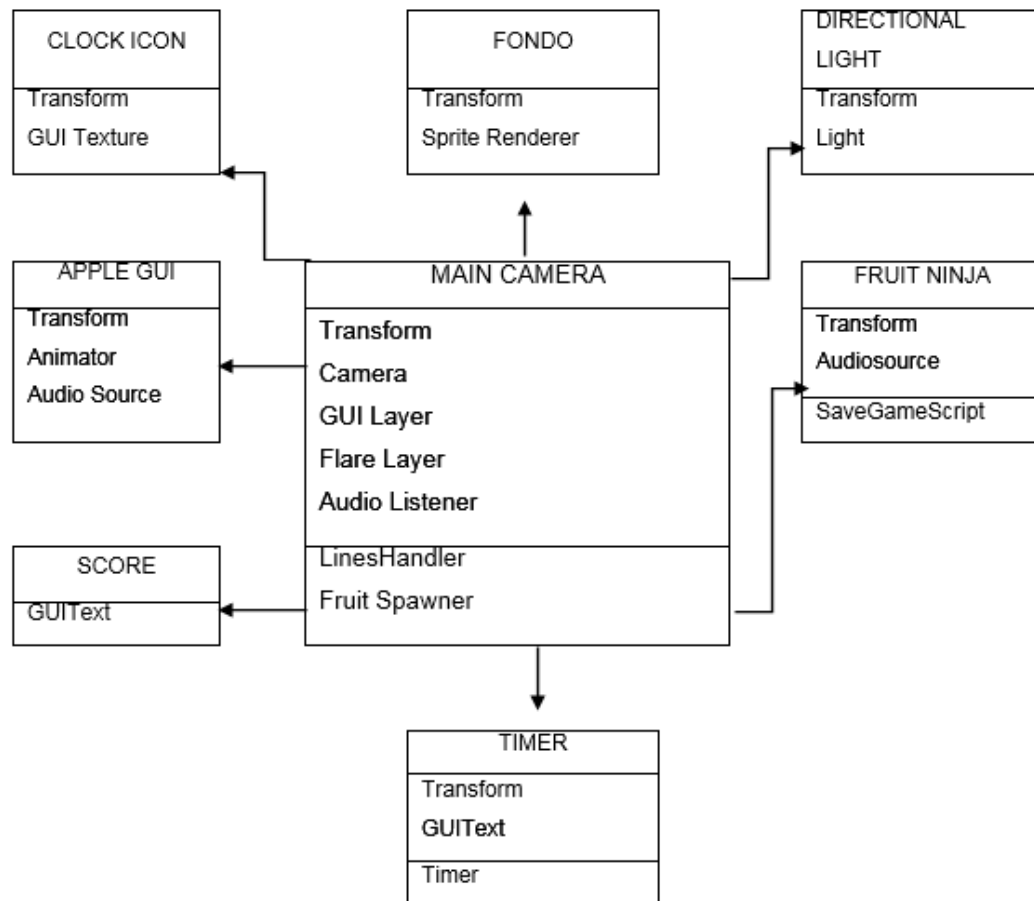


Figura III- 17. Diagrama de *GameObjects* del mini-juego *Fruit Ninja*. Fuente: Carlos Carvajal 2015.

Codificación

FruitSpawner.cs controla la creación de las rondas de manzanas. Se crea una ronda de 4 manzanas cada 6 segundos. Algunas variables de salida de cada manzana (ángulo x, y, z) son asignadas aleatoriamente de entre un rango predefinido. De esta manera aunque siempre se generan 4 manzanas, sus trayectorias no son siempre las mismas. Otras variables, como la gravedad, inercia, etc., son fijadas en el editor de *Unity*. La asignación de variables públicas se hace a través del editor de *Unity*. A continuación se muestra la figura III-18 la cual es una impresión de pantalla del código del mini-juego en *MonoDevelop*:

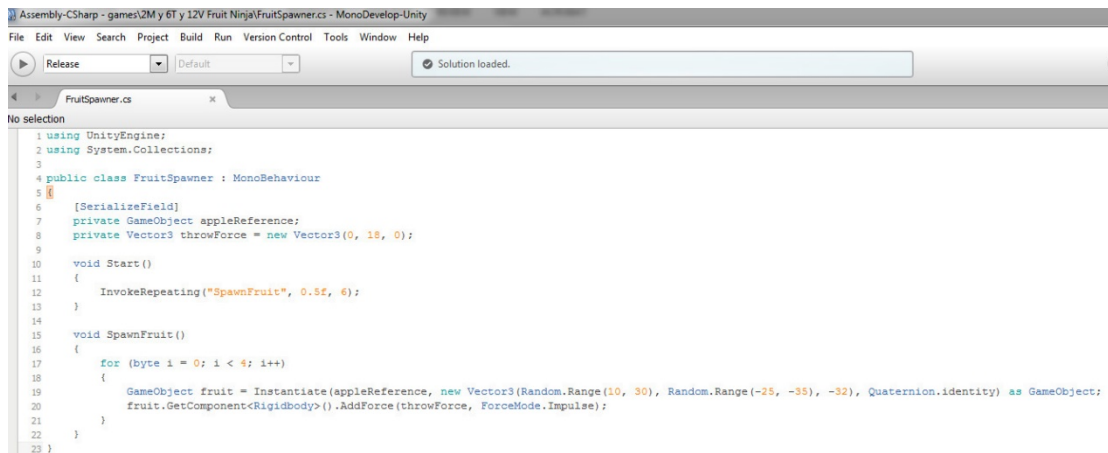


Figura III- 18. Impresión de pantalla del código del mini-juego *Fruit Ninja*, en *MonoDevelop*, de *Unity*.

Timer.cs controla el tiempo y las condiciones de fin de juego. *Apple.cs* controla todas las operaciones de instanciación de cada manzana. *LinesHandler.cs* controla todo lo relacionado a la interacción *touch* del corte de las manzanas. Esto se hace analizando la trayectoria del dedo contra la trayectoria de la manzana. Si hay colisión en un parámetro cercano de tiempo (*deltaTime*), se incrementa un punto y se activa la animación de la manzana aplastada según el código descrito en *Splash.cs*.

A continuación se muestra la figura III- 19 como ejemplo de vista del juego *Fruit Ninja* en el editor de *Unity*.

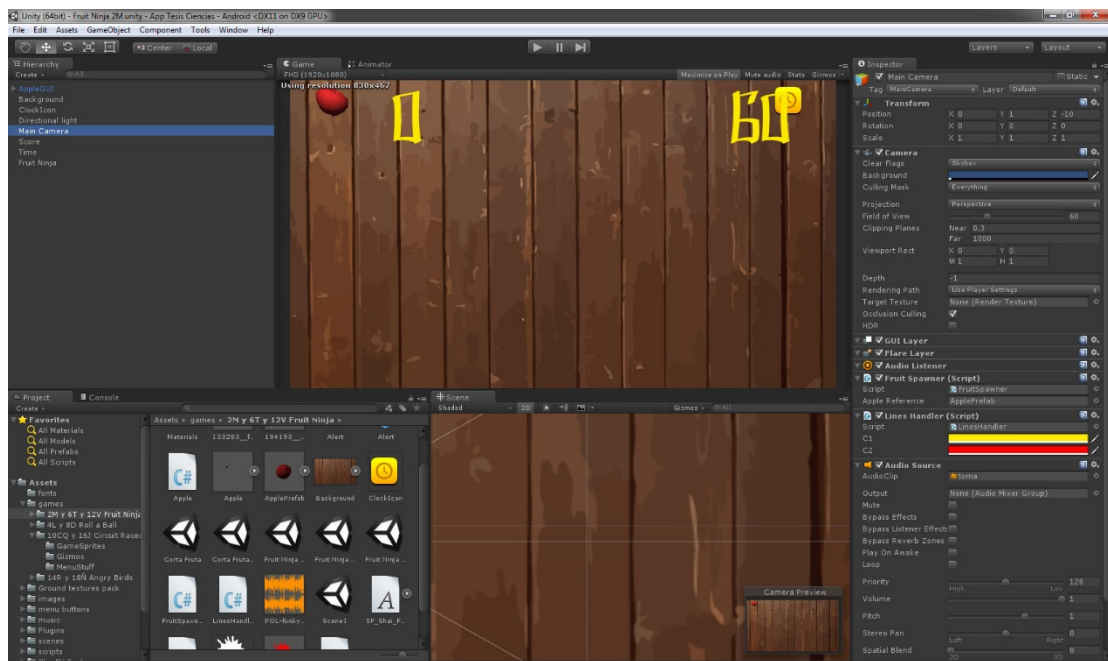


Figura III- 19. Impresión de pantalla del mini-juego *Ninja Fruit*, en el editor de Unity.

Pruebas

Se verifica que en efecto se cumplen todas las condiciones esperadas. Se presenta la imagen III-20 a modo de ejemplo:

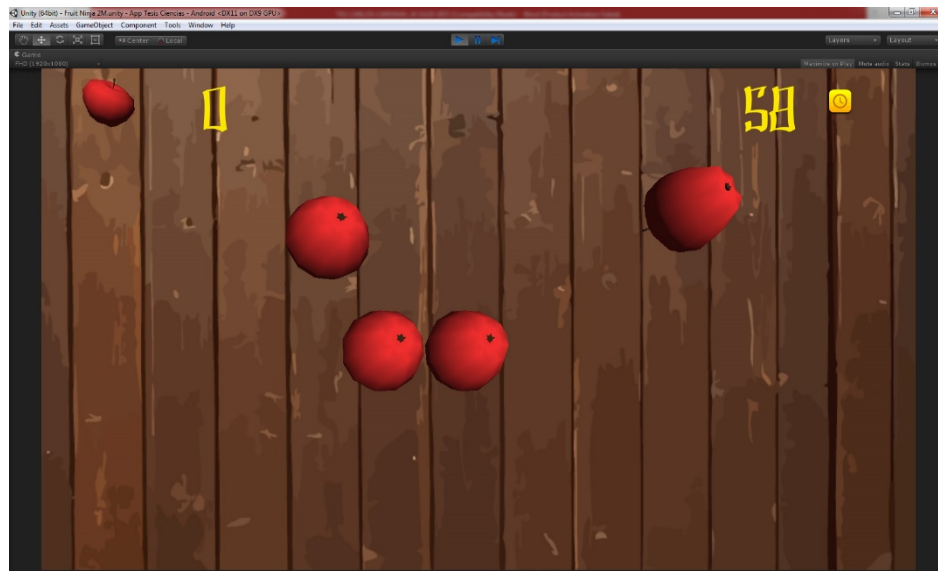


Figura III- 20. Impresión de pantalla del mini-juego *Ninja Fruit*, en el simulador de Unity.

3.14.2.2 *Roll a Ball*

El diseño de este mini-juego fue basado en otros juegos similares existentes. Se elige este tipo de juego ya que es lo suficientemente sencillo, fácil de usar y entender por cualquier niño a los que va dirigida esta aplicación y su método de control es muy innovador.

El juego consiste en hacer rodar la bola de manera que impacte contra todos los cubos con letras. Cada impacto suma un punto. El niño debe orientar la bola moviendo el dispositivo como si fuera la bola misma. Este juego hace uso de las funciones de acelerómetro integradas en los dispositivos *Android*.

Diagrama de clases (*GameObjects*)

A continuación se presenta la figura III- 21 la cual muestra el diagrama de *GameObjects* del mini-juego *Roll a Ball*. (Ver también las figuras [A-13](#), [A-14](#) y [A-15](#) en el [Apéndice A](#))

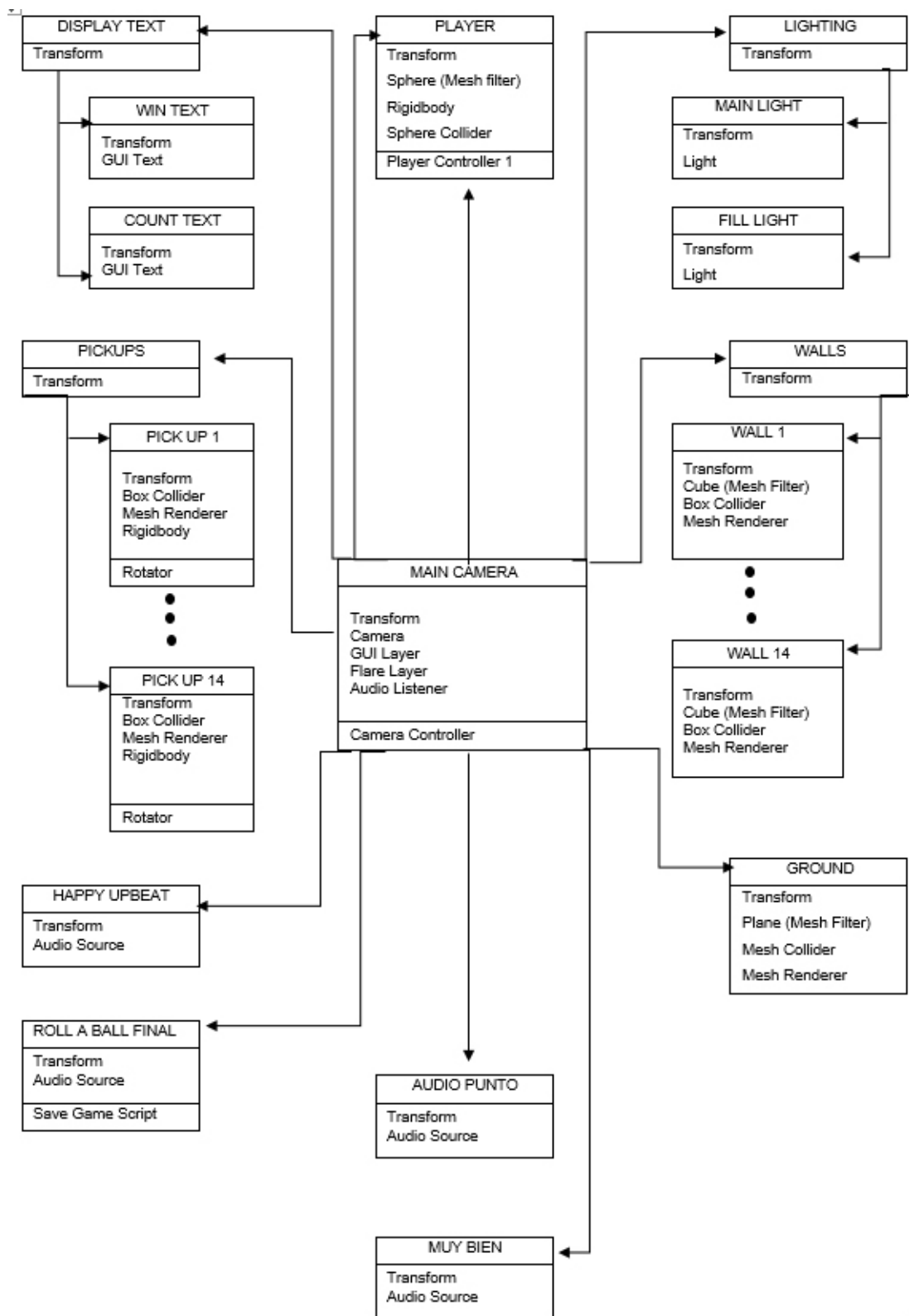
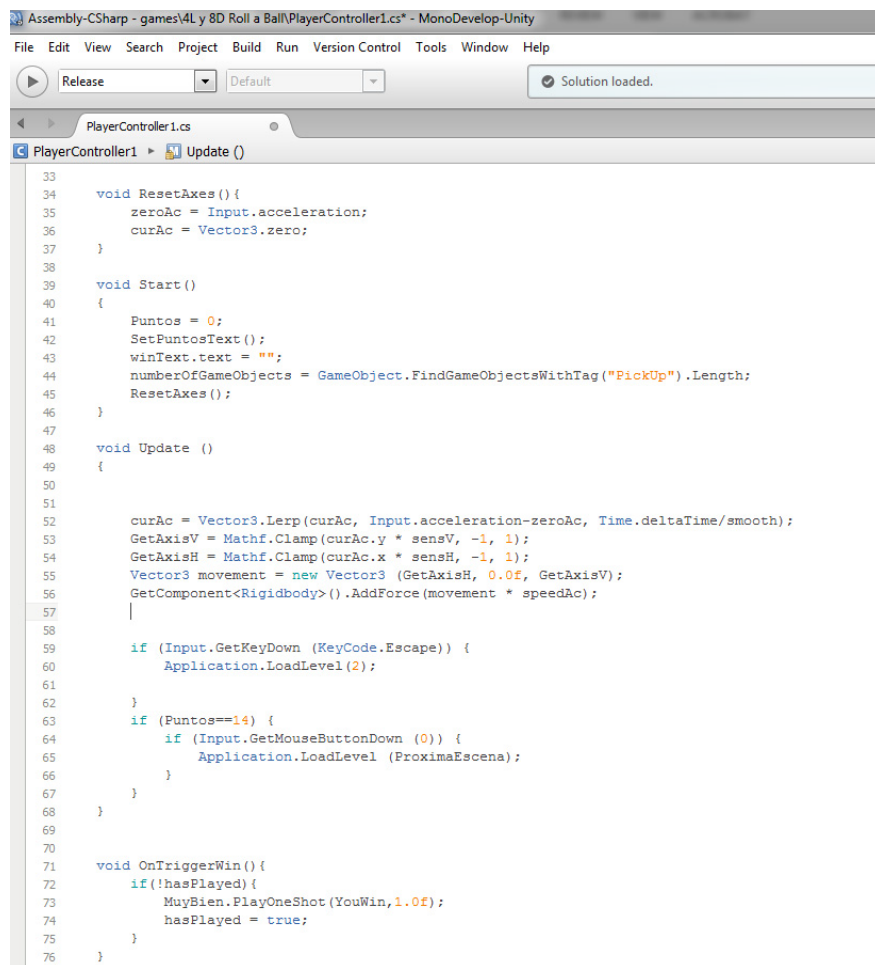


Figura III- 21. Diagrama de *GameObjects* del mini-juego *Roll a Ball*. Fuente: Carlos Carvajal 2015

Codificación

PlayerController1.cs controla el movimiento de la bola, asociado al cambio de los valores de los acelerómetros del dispositivo *Android*. En la imagen III-22 a continuación se puede observar como la función *Update()* actualiza la detección de los valores de los acelerómetros, a una velocidad de 1/60 segundos, y aplica las transformaciones al *GameObject Rigidbody* asociado al objeto bola.



```
Assembly-CSharp - games\4L y 8D Roll a Ball\PlayerController1.cs* - MonoDevelop-Unity
File Edit View Search Project Build Run Version Control Tools Window Help
Release Default Solution loaded.
PlayerController1.cs
PlayerController1 ▶ Update ()
33
34 void ResetAxes() {
35     zeroAc = Input.acceleration;
36     curAc = Vector3.zero;
37 }
38
39 void Start()
40 {
41     Puntos = 0;
42     SetPuntosText();
43     winText.text = "";
44     numberOfGameObjects = GameObject.FindGameObjectsWithTag("PickUp").Length;
45     ResetAxes();
46 }
47
48 void Update ()
49 {
50
51
52     curAc = Vector3.Lerp(curAc, Input.acceleration-zeroAc, Time.deltaTime/smooth);
53    .GetAxisV = Mathf.Clamp(curAc.y * sensV, -1, 1);
54    .GetAxisH = Mathf.Clamp(curAc.x * sensH, -1, 1);
55     Vector3 movement = new Vector3(GetAxisH, 0.0f, GetAxisV);
56     GetComponent<Rigidbody>().AddForce(movement * speedAc);
57
58
59     if (Input.GetKeyDown(KeyCode.Escape)) {
60         Application.LoadLevel(2);
61     }
62
63     if (Puntos==14) {
64         if (Input.GetMouseButtonDown(0)) {
65             Application.LoadLevel (ProximaEscena);
66         }
67     }
68 }
69
70 void OnTriggerWin(){
71     if(!hasPlayed){
72         MuyBien.PlayOneShot(YouWin,1.0f);
73         hasPlayed = true;
74     }
75 }
76 }
```

Figura III- 22. Impresión de pantalla del código *PlayerController1.cs** del mini-juego *Roll a Ball*, en *MonoDevelop* de *Unity*.

CameraFollow.cs mantiene el seguimiento y el comportamiento de la cámara sobre la bola. *Rotator.cs* controla la rotación de los cubos con letras. Esto se hace

incrementando los valores de rotación, en las opciones de transformación del objeto, cada 1/60 segundos.

A continuación se muestra la figura III-23 como un ejemplo de vista del juego *Roll a Ball* en el editor de *Unity*.

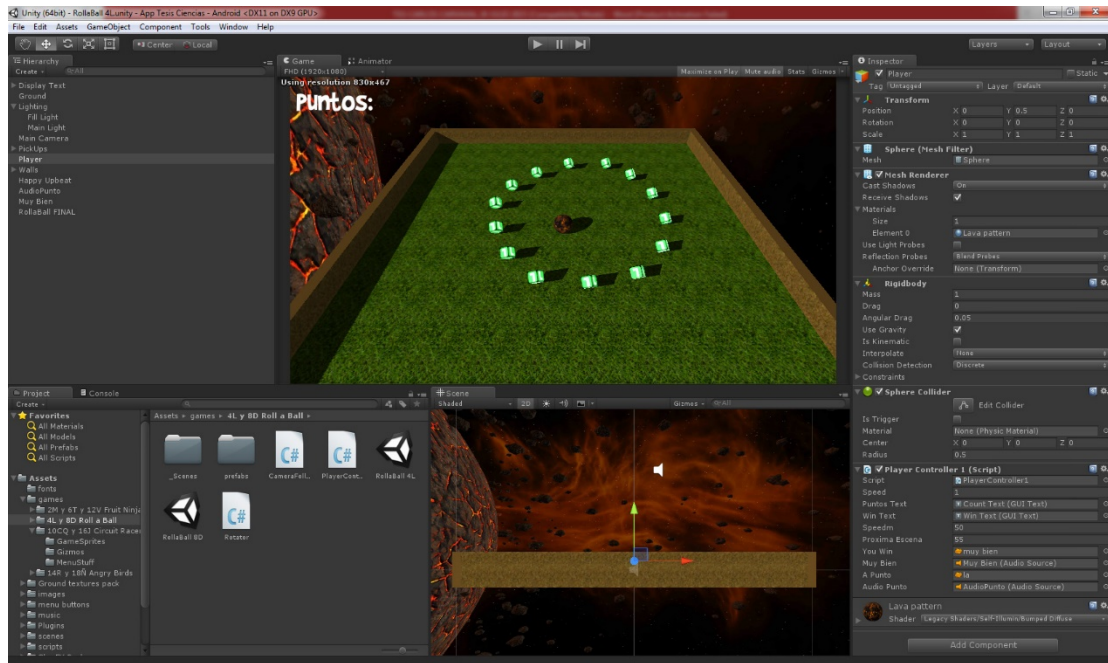


Figura III-23. Impresión de pantalla del mini-juego *Roll a Ball*, en el editor de *Unity*.

Pruebas

Se verifica que en efecto se cumplen todas las condiciones esperadas. Se presenta la imagen III-24 a modo de ejemplo:

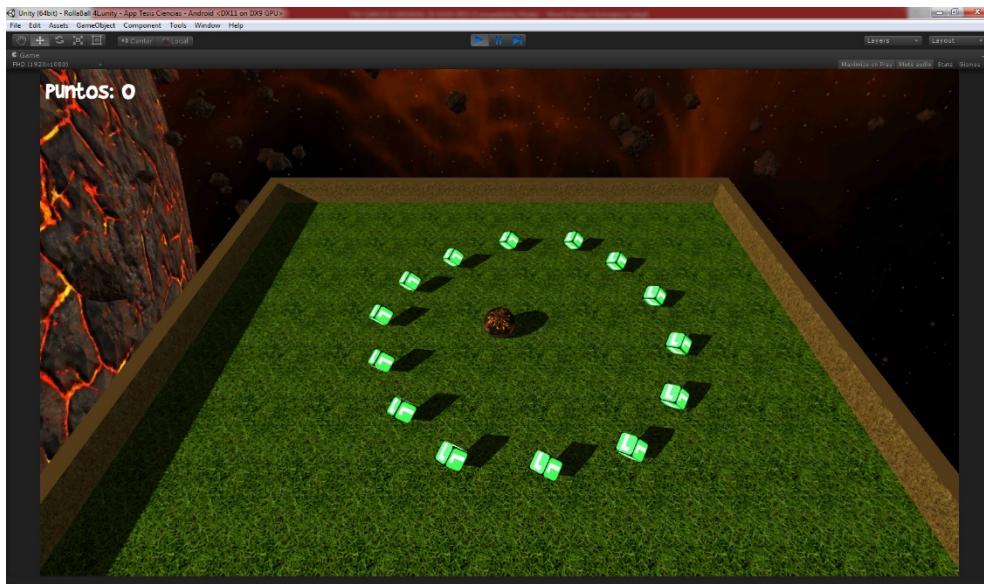


Figura III- 24. Impresión de pantalla del mini-juego *Roll a Ball*, en el simulador de *Unity*.

3.14.2.3 *Circuit Racer*

El diseño de este mini-juego fue basado en otros juegos similares existentes. Se elige este tipo de juego ya que es lo suficientemente sencillo, fácil de usar y entender por cualquier niño a los que va dirigida esta aplicación. Este juego se agrega más adelante en el curso de la aplicación ya que supone mucha mayor dificultad que los juegos anteriores, por el uso del control virtual.

El juego consiste en hacer rodar el carro de manera que de todas las vueltas necesarias por una pista cerrada, para ganar la carrera. El niño deberá elegir con cuál de los 3 carros ofrecidos desea hacer la carrera. Cada carro tiene una velocidad y una resistencia de giro del volante diferente.

La cantidad de vueltas y el tiempo requerido para darlas dependerá de la opción de dificultad elegida antes de iniciar el juego. El niño debe manejar el carro haciendo uso de un *stick* virtual sobre puesto en la parte inferior izquierda de la pantalla. Cabe destacar que gran parte de la dificultad del juego la hace el hecho de

que la vista del juego es desde arriba, y que se han añadido un par de cajas en la pista para entorpecer el paso del corredor.

Diagrama de clases (*GameObjects*)

Las figuras III-25, III-26, III-27 y III-28 a continuación muestran los diagramas de *GameObjects* para el mini-juego *Circuit Racer*. (Ver también las figuras [A-16](#), [A-17](#), [A-18](#), [A-19](#), [A-20](#) y [A-21](#) en el [Apéndice A](#))

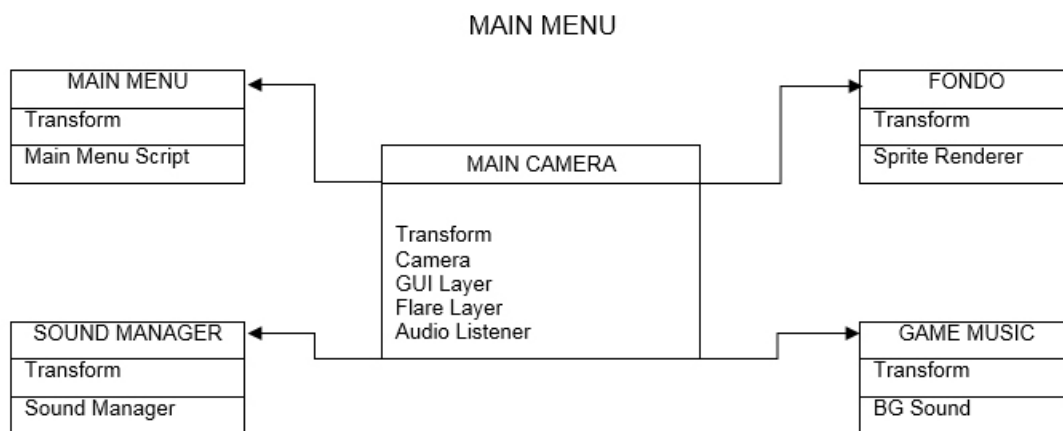


Figura III- 25. Diagrama de *GameObjects* *Main Menu* del mini-juego *Circuit Racer* Fuente: Carlos Carvajal 2015.

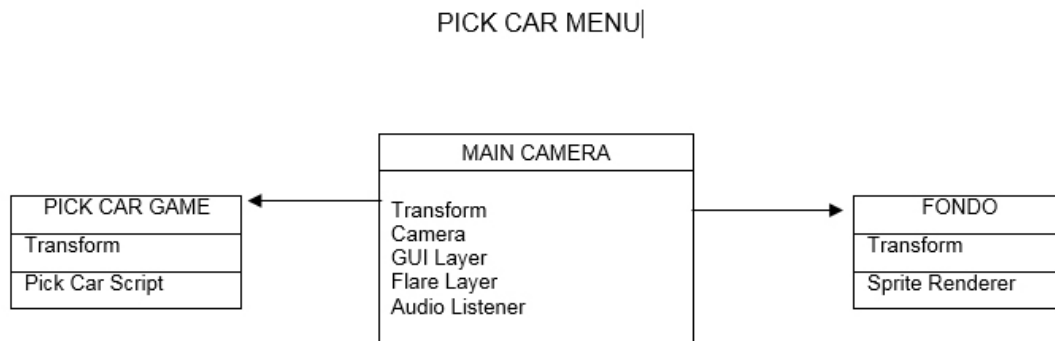


Figura III- 26. Diagrama de *GameObjects* *Pick Car Menu* del mini-juego *Circuit Racer* Fuente: Carlos Carvajal 2015.

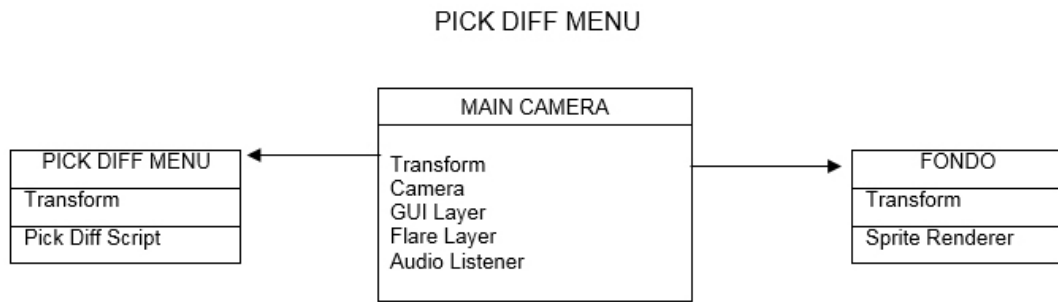


Figura III- 27. Diagrama de *GameObjects* *Pick Diff Menu* del mini-juego *Circuit Racer* Fuente: Carlos Carvajal 2015.

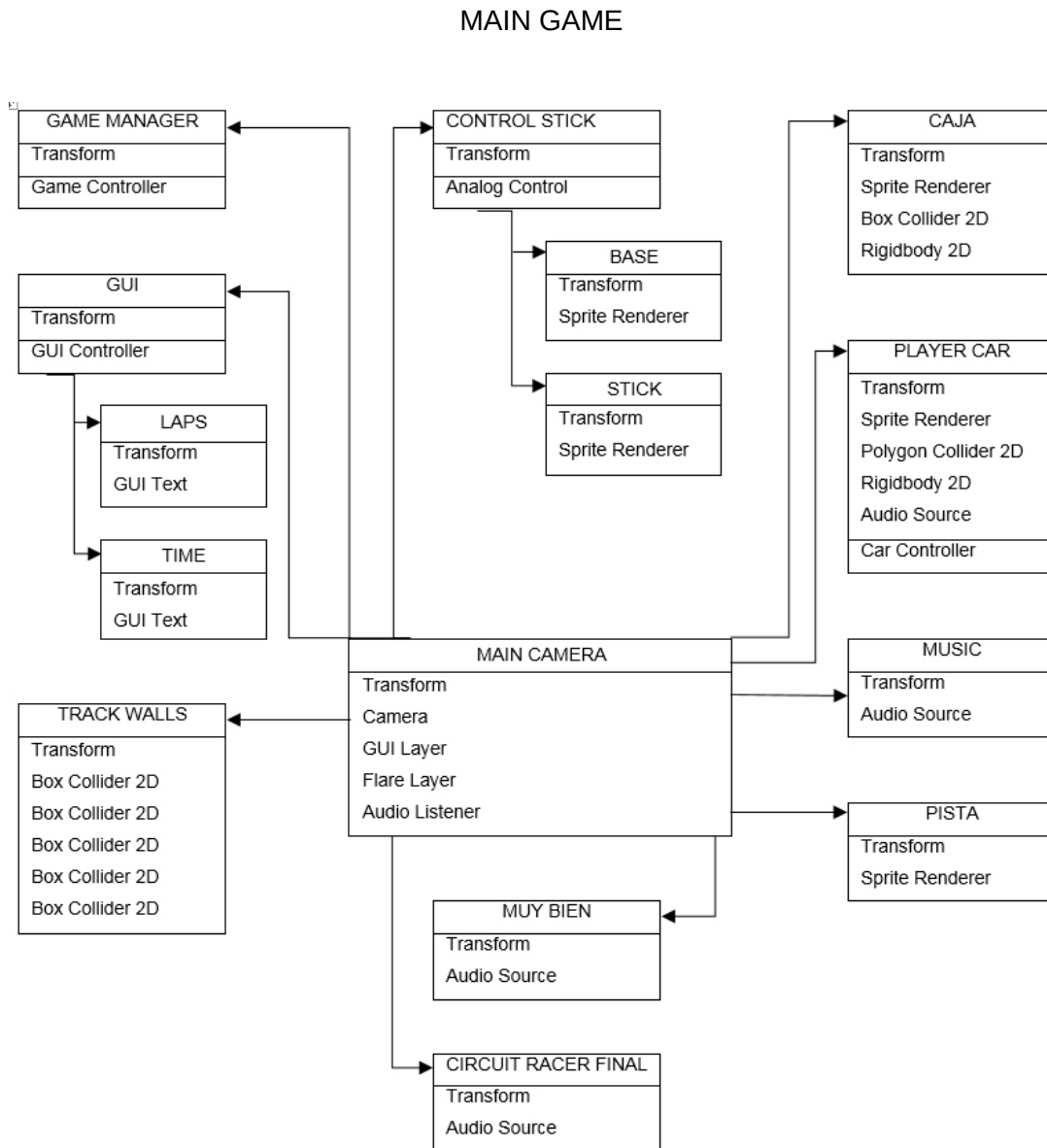
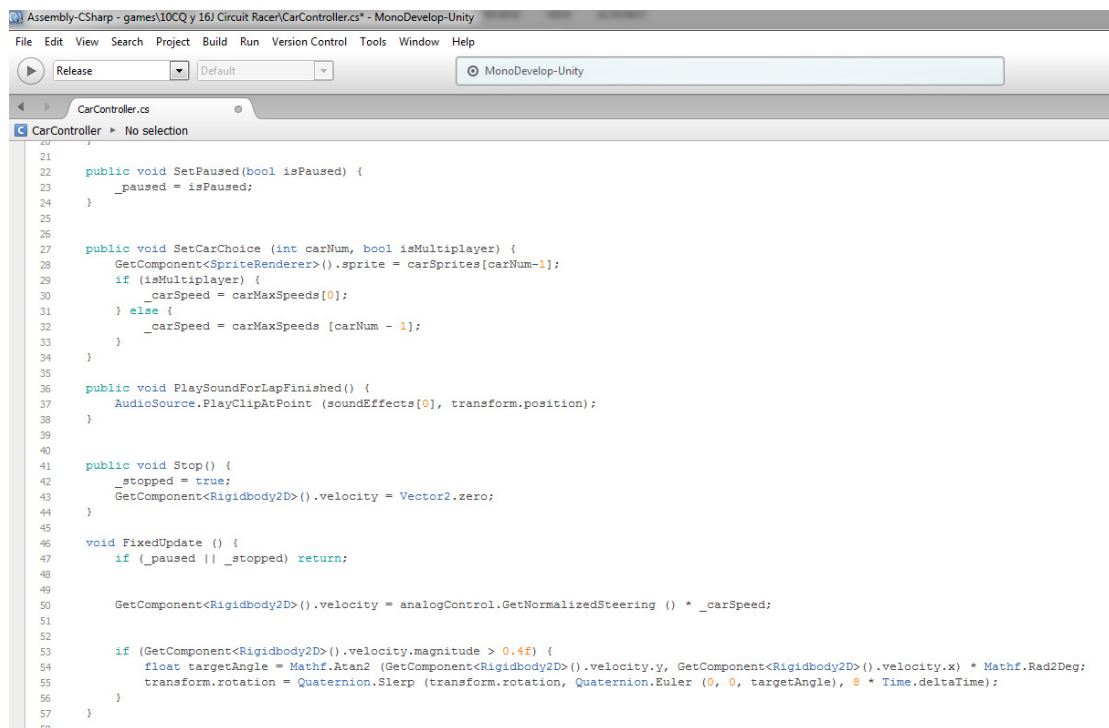


Figura III- 28. Diagrama de *GameObjects* del juego principal en el mini-juego *Circuit Racer*. Fuente: Carlos Carvajal 2015.

Codificación

CarController.cs maneja el movimiento del carro (con ayuda de *AnalogControl.cs* que maneja las propiedades del *stick* en sí mismas), asociando el movimiento del *stick* virtual a la transformación del objeto carro. En la imagen III-29 a continuación se puede observar como la función *FixedUpdate()* actualiza la detección de los valores del *stick*, a una velocidad de 1/30 segundos, y aplica las transformaciones al *GameObject Rigidbody* asociado al objeto carro.



```
21  
22 public void SetPaused(bool isPaused) {  
23     _paused = isPaused;  
24 }  
25  
26  
27 public void SetCarChoice (int carNum, bool isMultiplayer) {  
28     GetComponent<SpriteRenderer>().sprite = carSprites[carNum-1];  
29     if (isMultiplayer) {  
30         _carSpeed = carMaxSpeeds[0];  
31     } else {  
32         _carSpeed = carMaxSpeeds [carNum - 1];  
33     }  
34 }  
35  
36 public void PlaySoundForLapFinished () {  
37     AudioSource.PlayClipAtPoint (soundEffects[0], transform.position);  
38 }  
39  
40  
41 public void Stop () {  
42     _stopped = true;  
43     GetComponent<Rigidbody2D>().velocity = Vector2.zero;  
44 }  
45  
46 void FixedUpdate () {  
47     if (_paused || _stopped) return;  
48  
49  
50     GetComponent<Rigidbody2D>().velocity = analogControl.GetNormalizedSteering () * _carSpeed;  
51  
52  
53     if (GetComponent<Rigidbody2D>().velocity.magnitude > 0.4f) {  
54         float targetAngle = Mathf.Atan2 (GetComponent<Rigidbody2D>().velocity.y, GetComponent<Rigidbody2D>().velocity.x) * Mathf.Rad2Deg;  
55         transform.rotation = Quaternion.Slerp (transform.rotation, Quaternion.Euler (0, 0, targetAngle), 8 * Time.deltaTime);  
56     }  
57 }  
58
```

Figura III- 29. Impresión de pantalla del código *CarController.cs** del mini-juego *Circuit Racer*, en *MonoDevelop de Unity*.

GameController.cs mantiene el seguimiento de las variables de selección de carro y de dificultad (haciendo uso a su vez de *RetainedUserPickScript.cs*) y verifica los estados de juego (tiempo, vueltas). *GUIController.cs* se utiliza para la personalización de botones en los menús y en el estado del juego. Por último *BGSound.cs* controla la música de fondo de manera continua a través de los menús del juego.

A continuación se muestra la figura III-30 como un ejemplo de la vista del juego *Circuit Racer* en el editor de Unity.

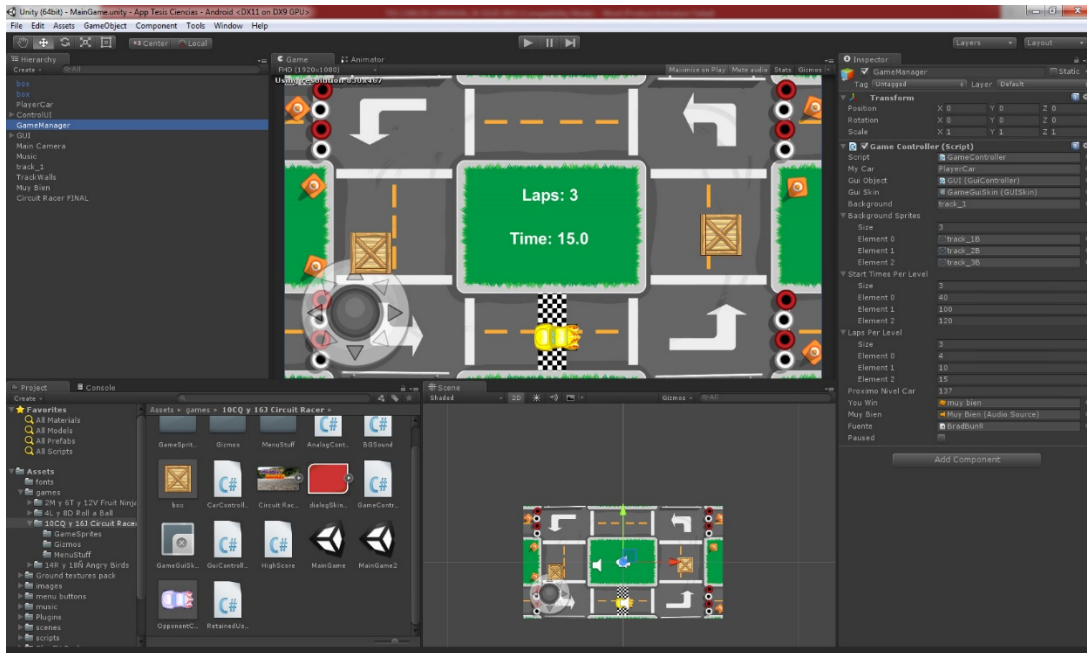


Figura III- 30. Impresión de pantalla del mini-juego *Circuit Racer*, en el editor de Unity.

Pruebas

Se verifica que en efecto se cumplen todas las condiciones esperadas. Se presenta la imagen III-31 como ejemplo:



Figura III- 31. Impresión de pantalla del mini-juego *Circuit Racer*, en el simulador de Unity.

3.14.2.3 Angry Letters

El diseño de este mini-juego fue basado en otros juegos similares existentes. Se elige este tipo de juego ya que es lo suficientemente sencillo, fácil de usar y entender por cualquier niño a los que va dirigida esta aplicación. Este juego se agrega más adelante en el curso de la aplicación ya que supone mucha mayor dificultad que los juegos anteriores, debido a los cálculos de trayectoria necesarios para dominarlo. Sin embargo, el nivel desarrollado representa una dificultad mínima, una vez se dominen los controles básicos.

El juego consiste en utilizar una honda para lanzar hasta tres bolas de madera, con el objetivo de hacerlas impactar contra los 3 cubos de letras dispuestos en el escenario. Tal como en el caso de uso de una honda real, el niño debe halarla hacia atrás, ajustar la fuerza y el ángulo de lanzamiento y luego soltar la honda para que la bola salga disparada hacia el objetivo. En este caso, el niño no utilizará su mano, sino su dedo.

Adicionalmente, los cubos de letras están encerrados en estructuras de madera de alta fragilidad, por lo que al ser impactadas por la bola, estas estructuras reaccionan al impacto, la resistencia y la gravedad de maneras muy realistas. Se ha diseñado una física tan realista que incluso los golpes de estas estructuras contra los cubos de letras las pueden destruir, si la fuerza aplicada en contra logra ser suficiente. La mecánica es tremendamente simple y adictiva, lo que resulta en un juego muy divertido.

Diagrama de clases (*GameObjects*)

A continuación se presenta la figura III-32 como diagrama del mini-juego *Angry Letters*. (Ver también las figuras [A-22](#), [A-23](#), [A-24](#), [A-25](#), [A-26](#), [A-27](#), [A-28](#), [A-29](#), [A-30](#), [A-31](#), [A-32](#), [A-33](#) y [A-34](#) en el [Apéndice A](#))

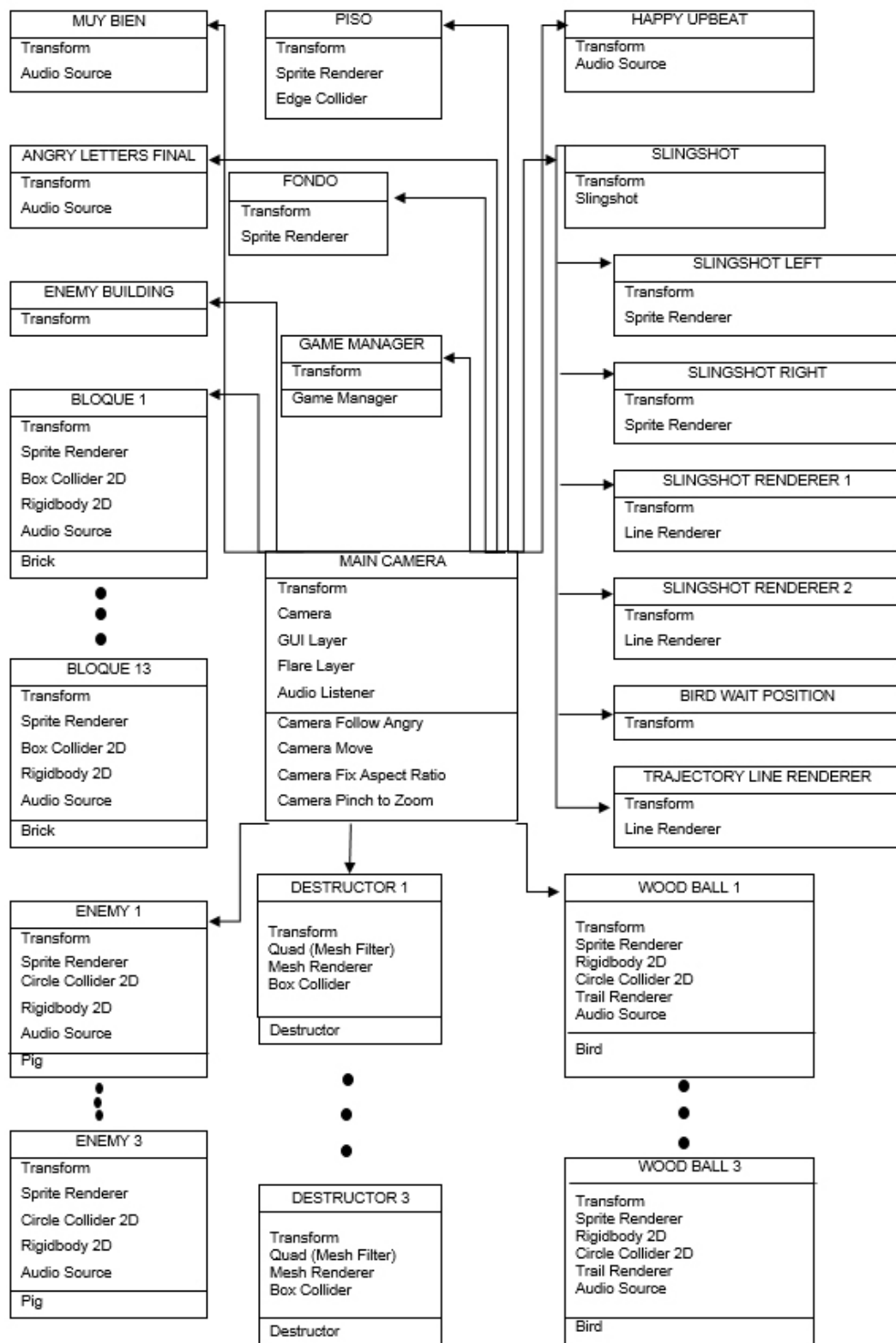


Figura III- 32. Diagrama de GameObjects del mini-juego *Angry Letters*. Fuente: Carlos Carvajal 2015.

Codificación

GameManager.cs controla a nivel general las operaciones de estado del juego. Para tal fin se utilizan muchos *scripts* auxiliares. *Bird.cs* controla los aspectos de las bolas de madera tales como trayectoria de vuelo (transformaciones asociadas), tiempo de vida, colisiones y otros. *Brick.cs* se encarga de la física e interacciones asociadas (colisiones) a los elementos de las estructuras de madera donde se encuentran los cubos de letras.

CameraController.cs se encarga de las operaciones de vistas y cámara. Este *script* se apoya en *CameraFixAspectRatio.cs* (mantiene la corrección de aspecto de pantalla), *CameraFollowAngry.cs* (permite que una vez lanzada la bola, la cámara la siga), *CameraMove.cs* (establece los patrones de movimiento de la cámara) y *CameraPinchToZoom.cs* (permite el uso de dos dedos para aumentar o reducir el zoom de toda la escena de juego) para realizar determinadas funciones.

Constants.cs especifica ciertos valores constantes a lo largo de todo el juego, como radios de colisión de las bolas de madera y la velocidad mínima de vuelo de la bola. *Destroyer.cs* elimina todo lo que salga de la escena de juego. *Enums.cs* controla todos los tiempos de espera y los cambios de estado de los objetos.

Pigs.cs controla la respuesta a la colisiones contra los cubos de letras. Y por último, *SlingShot.cs* se encarga de controlar todo el comportamiento de la honda y de la proyección de lanzamiento, y el lanzamiento, de las bolas de madera.

A continuación se muestra la figura III-33 como ejemplo de vista del juego *Angry Letters* en el editor de *Unity*, donde se pueden apreciar los valores de las variables públicas de *SlingShot.cs*:

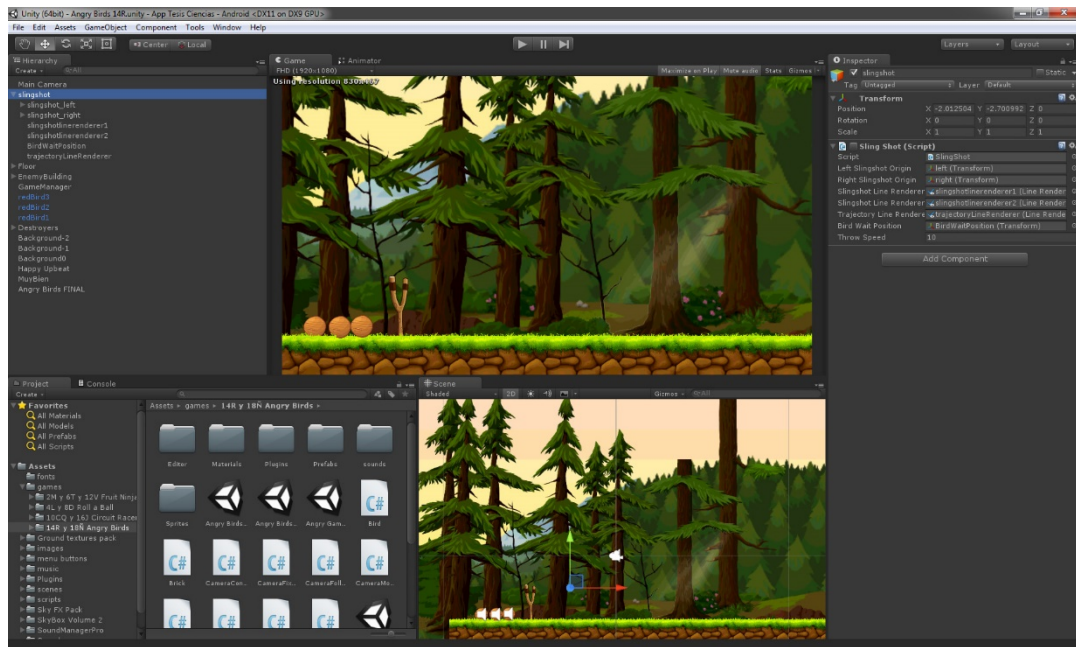


Figura III- 33. Impresión de pantalla del mini-juego *Angry Letters*, en el editor de Unity.

Pruebas

Se verifica que en efecto se cumplen todas las condiciones esperadas. Se presenta la imagen III-34 de ejemplo:

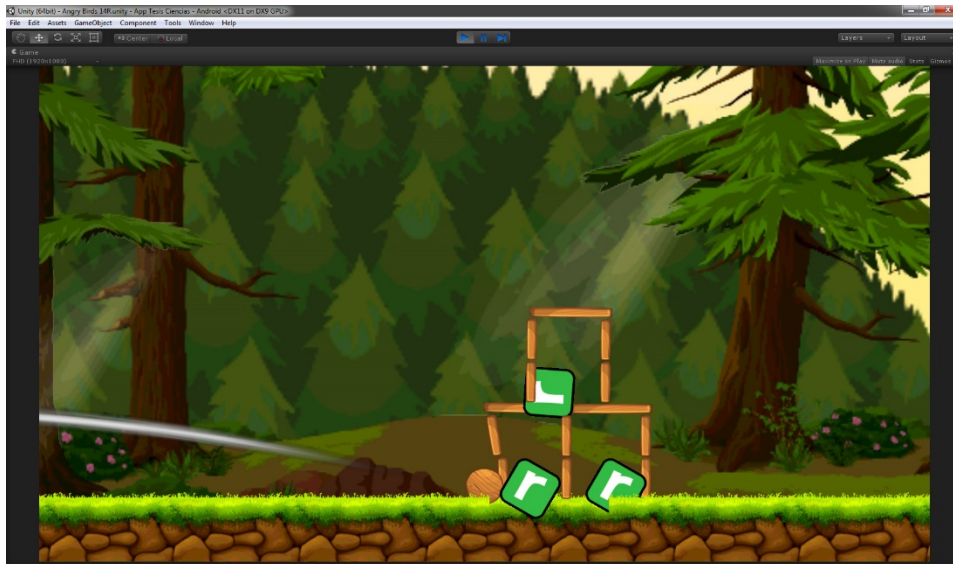


Figura III- 34. Impresión de pantalla del mini-juego *Angry Letters*, en el simulador de Unity.

3.15 Cuarta iteración

A continuación se lista la documentación de la planificación, diseño, codificación y pruebas que tuvieron lugar durante la cuarta iteración del proyecto.

3.15.1 Planificación

A continuación se lista la documentación de la planificación de la cuarta iteración

3.15.1.1 Historias de usuario

Historia de Usuario	
Identificador: 13	Nombre: Menú Principal
Usuario: Usuario	Prioridad: Alta
Programador: Carlos Carvajal	
Descripción: el usuario ingresa al menú principal luego de pasar pantallas de Bienvenida. Allí debe seleccionar entre las 5 opciones disponibles (Aprende ya, Menú de clases, Salir, Créditos y Reiniciar app).	

Historia de Usuario	
Identificador: 14	Nombre: Aprende ya
Usuario: Usuario	Prioridad: Alta
Programador: Carlos Carvajal	
Descripción: el usuario empieza el curso, dirigiéndose automáticamente a la primera actividad de la primera clase.	

Historia de Usuario	
Identificador: 15	Nombre: Continuar
Usuario: Usuario	Prioridad: Alta

Programador: Carlos Carvajal

Descripción: en el caso de que el usuario tenga al menos una clase terminada (y guardada), el texto del botón deberá cambiar a “Continuar...” y al presionarlo, debe continuar el curso en la primera actividad de la clase siguiente a la última completada.

Historia de Usuario

Identificador:16

Nombre: Salir

Usuario: Usuario

Prioridad: Alta

Programador: Carlos Carvajal

Descripción: el usuario utilizará este botón para salir de manera segura de la app.

Historia de Usuario

Identificador:17

Nombre: Créditos

Usuario: Usuario

Prioridad: Baja

Programador: Carlos Carvajal

Descripción: en esta opción el usuario podrá ver información acerca de las personas involucradas en la creación de esta aplicación.

Historia de Usuario

Identificador:18

Nombre: Reiniciar app

Usuario: Usuario

Prioridad: Alta

Programador: Carlos Carvajal

Descripción: con esta opción el usuario podrá reiniciar el avance del curso. Debe generarse un método que evite el borrado accidental del progreso, por parte del usuario.

Historia de Usuario

Identificador: 19	Nombre: Menú de clases
Usuario: Usuario	Prioridad: Alta
Programador: Carlos Carvajal	
Descripción: esta opción le permite al usuario iniciar la clase que desee, entre las que ha aprobado hasta el momento.	

3.15.1.2 Plan de entrega

Luego de reunirse todos los actores del proyecto, se determina que para la primera entrega se desarrollarían las funcionalidades genéricas de las historias de usuario desde la 13 hasta la 19.

3.15.1.3 Velocidad del proyecto

Se estima un tiempo de desarrollo de siete (07) días hábiles para la realización de las historias de usuario, desde la historia número 13 hasta la número 19.

3.15.1.4 Tareas específicas

Tarea 13	
Identificador: 13	Número de historia: 13
Nombre: Menú principal.	Tipo: Alta
Inicio: 06/07/2015	Fin: 07/07/2015
Responsable: Carlos Carvajal	
Descripción: el usuario ingresa al menú principal luego de pasar pantallas de copyright. Allí debe seleccionar entre las 5 opciones disponibles (Aprende ya, Menú de clases, Salir, Créditos y Reiniciar app).	
Pruebas de aceptación: - Verificar que la apariencia del menú sea adecuada. - Verificar que el audio funcione adecuadamente. - Verificar que los botones de opciones funcionen adecuadamente.	

Tarea 14	
Identificador: 14	Número de historia: 14
Nombre: Aprende ya.	Tipo: Alta
Inicio: 07/07/2015	Fin: 08/07/2015
Responsable: Carlos Carvajal	
Descripción: el usuario empieza el curso, dirigiéndose automáticamente a la primera actividad de la primera clase.	
Pruebas de aceptación: - Verificar que la apariencia del menú sea adecuada. - Verificar que el audio funcione adecuadamente. - Verificar que el botón se comporte de manera adecuada.	

Tarea 15	
Identificador: 15	Número de historia: 15
Nombre: Continuar...	Tipo: Alta
Inicio: 07/07/2015	Fin: 08/07/2015
Programador: Carlos Carvajal	
Descripción: en el caso de que el usuario tenga al menos una clase terminada (y guardada), el texto del botón deberá cambiar a "Continuar..." y al presionarlo, debe continuar el curso en la primera actividad de la clase siguiente a la última completada.	
Pruebas de aceptación: - Verificar que la apariencia del menú sea adecuada. - Verificar que el audio funcione adecuadamente. - Verificar que el botón se comporte de manera adecuada.	

Tarea 16	
Identificador: 16	Número de historia: 16

Nombre: Salir.	Tipo: Alta
Inicio: 07/07/2015	Fin: 08/07/2015
Programador: Carlos Carvajal	
Descripción: el usuario utilizará este botón para salir de manera segura de la app.	
Pruebas de aceptación: - Verificar que la apariencia del menú sea adecuada. - Verificar que el audio funcione adecuadamente. - Verificar que el botón se comporte de manera adecuada (salida por doble toque de la tecla retroceder y salida presionando el botón “Salir del juego”).	

Tarea 17	
Identificador: 17	Número de historia: 17
Nombre: Créditos.	Tipo: Alta
Inicio: 08/07/2015	Fin: 09/07/2015
Programador: Carlos Carvajal	
Descripción: en esta opción el usuario podrá ver información acerca de las personas involucradas en la creación de esta aplicación.	
Pruebas de aceptación: - Verificar que la apariencia del menú y la información mostrada sea adecuada. - Verificar que el audio funcione adecuadamente. - Verificar que el botón “Regresar” se comporte de manera adecuada.	

Tarea 18	
Identificador: 18	Número de historia: 18
Nombre: Reiniciar app.	Tipo: Alta
Inicio: 09/07/2015	Fin: 10/07/2015
Programador: Carlos Carvajal	
Descripción: con esta opción el usuario podrá reiniciar el avance del curso. Debe generarse un método que evite el borrado accidental del progreso, por parte del usuario.	
Pruebas de aceptación:	

- Verificar que la apariencia del menú sea adecuada.
- Verificar que el audio funcione adecuadamente.
- Verificar que el botón se comporte de manera adecuada.
- Verificar que el procedimiento de palabra clave funcione adecuadamente para cada caso (caso de palabra correcta y caso de palabra incorrecta).

Tarea 19	
Identificador: 19	Número de historia: 19
Nombre: Menú de clases.	Tipo: Alta
Inicio: 13/07/2015	Fin: 14/07/2015
Programador: Carlos Carvajal	
Descripción: esta opción le permite al usuario iniciar la clase que desee, entre las que ha aprobado hasta el momento.	
Pruebas de aceptación: - Verificar que la apariencia del menú sea adecuada. - Verificar que el audio funcione adecuadamente. - Verificar que los botones se comporten de manera adecuada. - Verificar que se habiliten y deshabiliten las clases de manera adecuada. - Verificar que el mensaje de error se muestre adecuadamente.	

3.15.1.5 Reuniones

Durante esta primera iteración se llevaron a cabo reuniones cortas y de manera regular, en las que se discutían las tareas a desarrollar, problemas que se podrían presentar, o que se hayan presentado, y sus posibles soluciones. La duración de estas reuniones estuvo comprendida entre quince (15) y treinta (30) minutos.

3.15.2 Diseño

Una vez organizado el esquema de menús, se procedió a decidir el aspecto gráfico y la música adecuada para los mismos.

3.15.2.1 Diagramas de Clases (*GameObjects*)

A continuación se presentan las figuras III-35, III-36, III-37, III-38 y III-39 que muestra los diagramas de clase del Menú Principal de la aplicación. (Ver también las figuras [A-35](#), [A-36](#), [A-37](#), [A-38](#), [A-39](#), [A-40](#), [A-41](#), [A-42](#), [A-43](#), [A-44](#) y [A-45](#) en el [Apéndice A](#))

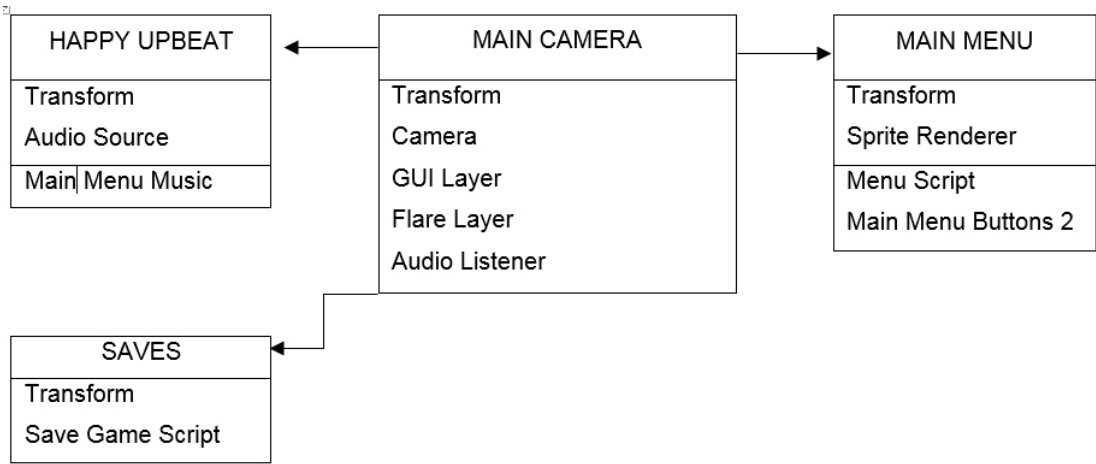


Figura III- 35. Diagrama de GameObjects del menú principal. Fuente: Carlos Carvajal 2015.

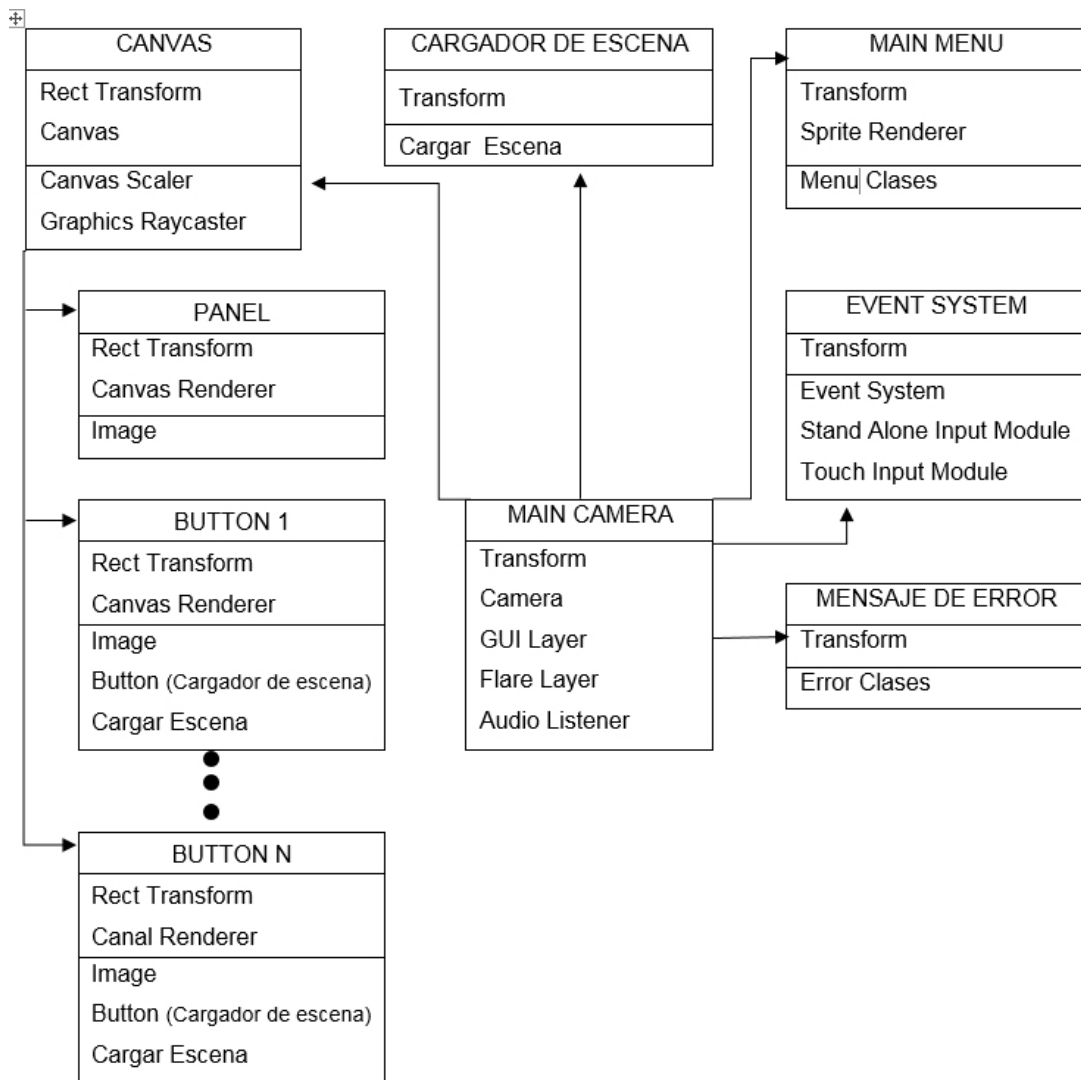


Figura III- 36. Diagrama de GameObjects del menú "Seleccionar clase". Fuente: Carlos Carvajal 2015.

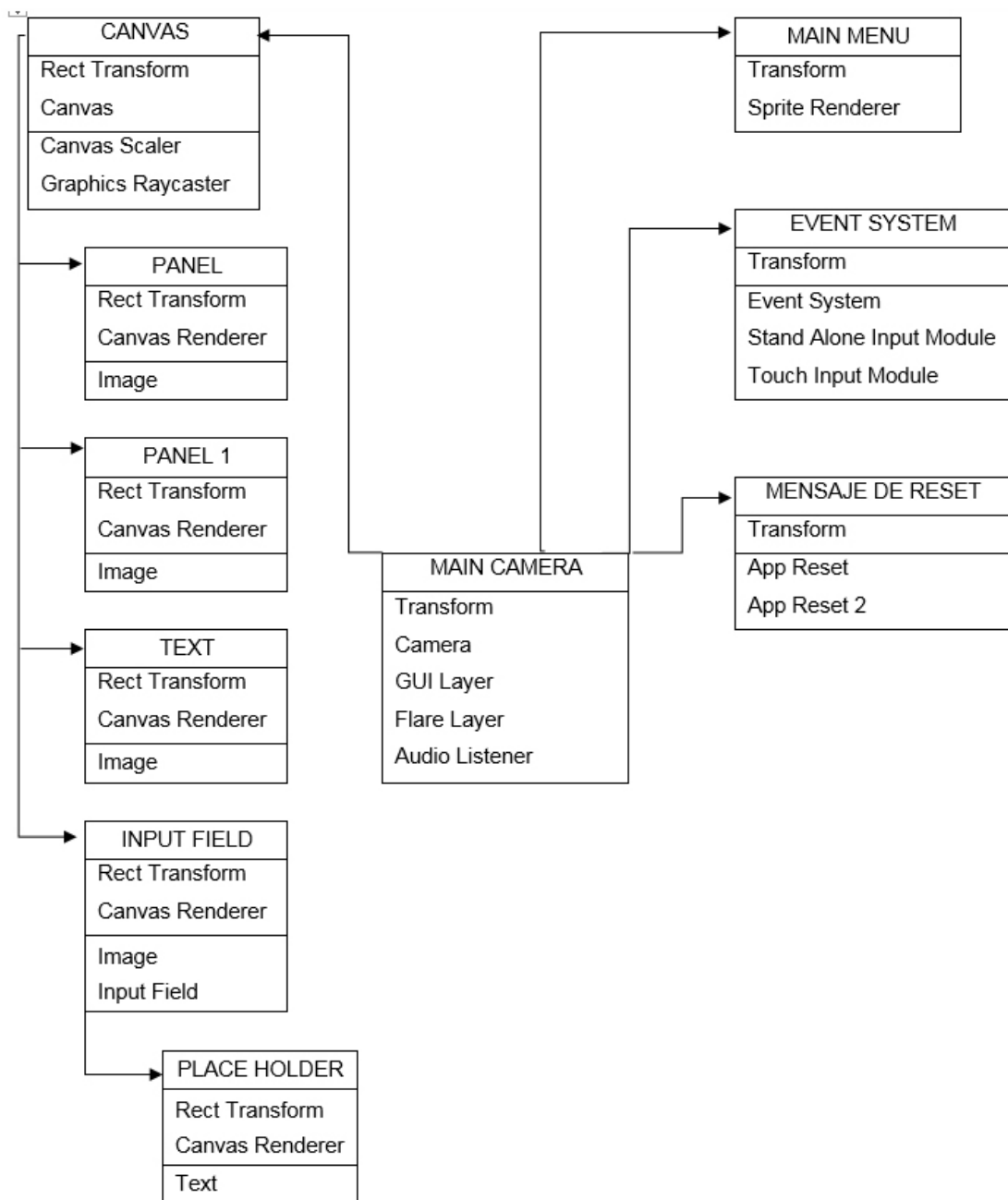


Figura III- 37. Diagrama de GameObjects del menú "Reiniciar app". Fuente: Carlos Carvajal 2015.

Créditos

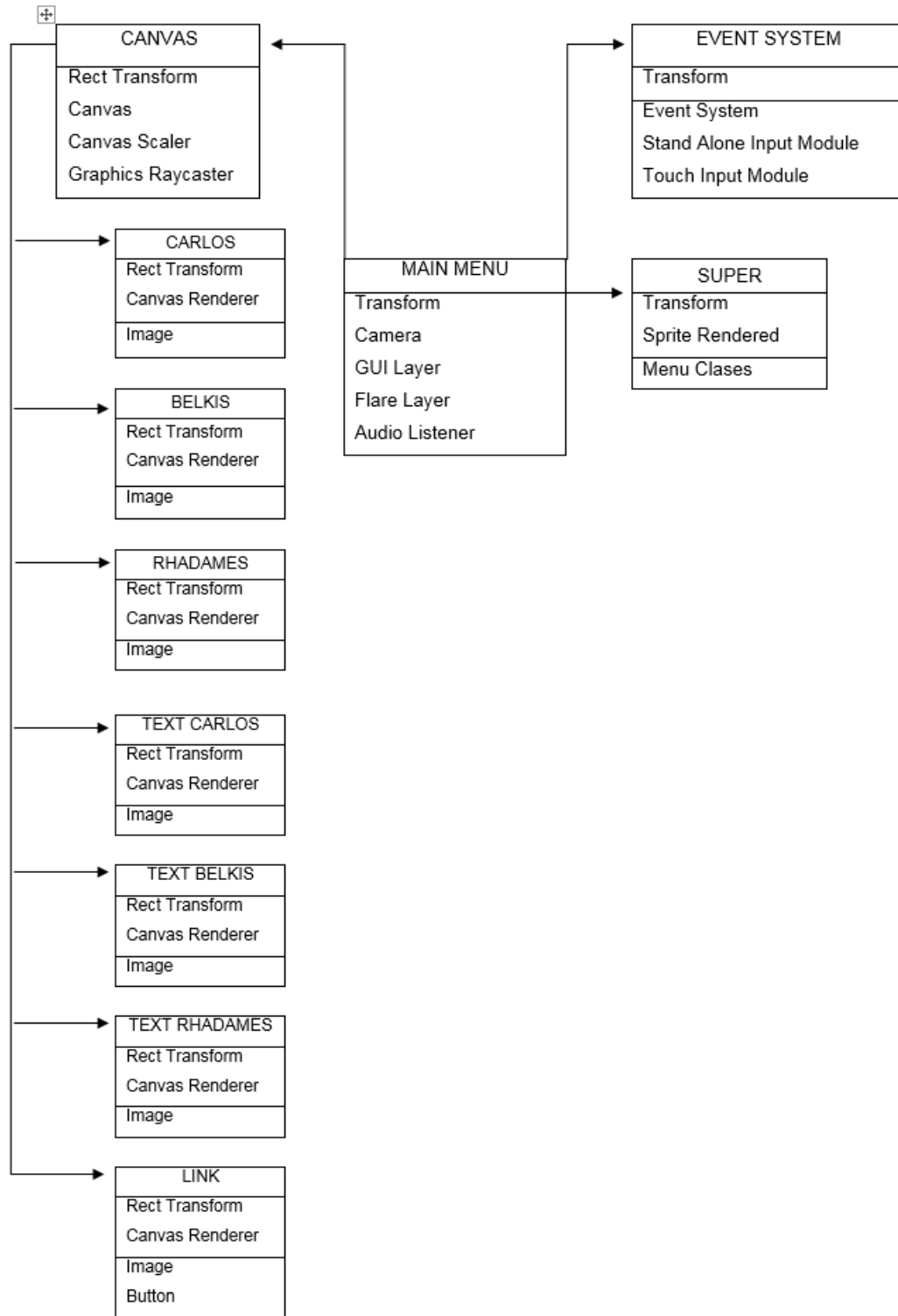


Figura III- 38. Diagrama de GameObjects del menú "créditos". Fuente: Carlos Carvajal 2015.

Disclaimer

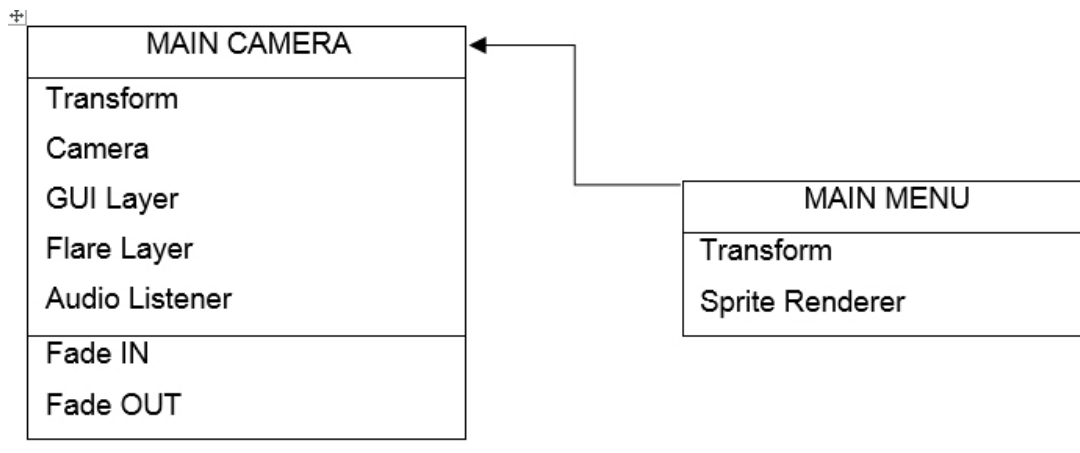


Figura III- 39. Diagrama de GameObjects de la pantalla "Disclaimer". Fuente: Carlos Carvajal 2015.

3.15.2.3 Codificación

Primero fue necesario dividir la tarea por menús. Se creó un *script* para manejar cada actividad en particular, y dado que se desea mantener el mismo fondo musical a lo largo de los mismos (sin interrupciones, ni reinicio), se creó también el *script* *MainMenuMusic.cs* para tal fin.

Un de las tareas mas complejas en esta etapa fué crear una opción que permitiera reinicar la aplicación, de tal manera que no fuese excesivamente complicada para el usuario, pero en la que a la vez se evitara la activación de la misma de manera accidental. Por esta razón se creó el método de reinicio por palabra clave.

La otra tarea compleja fué un menú de clases con autobloqueo de clases no completadas. Este menú le permite al niño repasar una clase anterior, pero le impide ver una clase que no haya "desbloqueado" completándola.

Ambas desafíos fueron solventados de manera sencilla. El primero a través del uso del método *InputField()* y el segundo utilizando el método *PlayerPrefs()*.

3.15.2.4 Pruebas

Durante esta etapa se realizaron una serie de pruebas utilizando el editor de *Unity*, basadas en las pruebas de aceptación especificadas en las tareas de desarrollo. A continuación se puede observar en las figuras III-40, III-41, III-42, III-43, III-44 y III-46 la aprobación de todas las pruebas (de un total de seis [6] pruebas) al finalizar esta iteración:

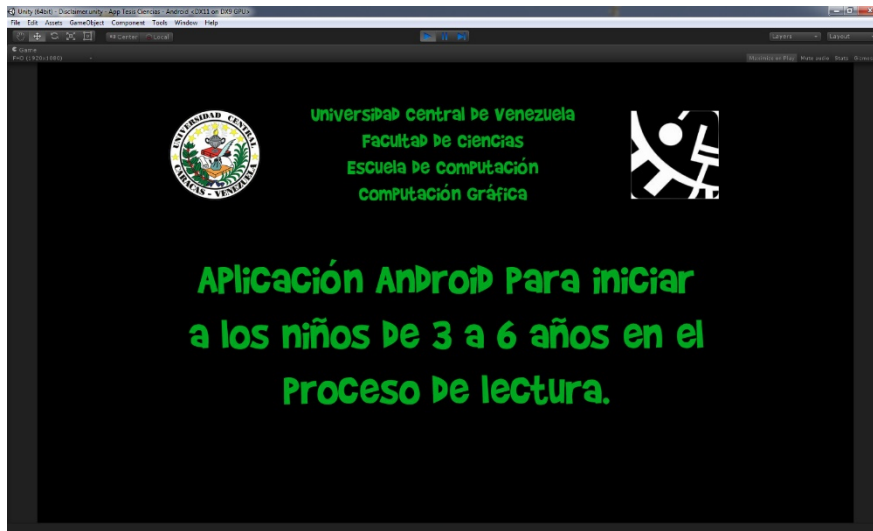


Figura III- 40. Impresion de pantalla de prueba de la pantalla “Disclaimer”.



Figura III- 41. Impresion de pantalla de prueba del menú principal de la aplicación.

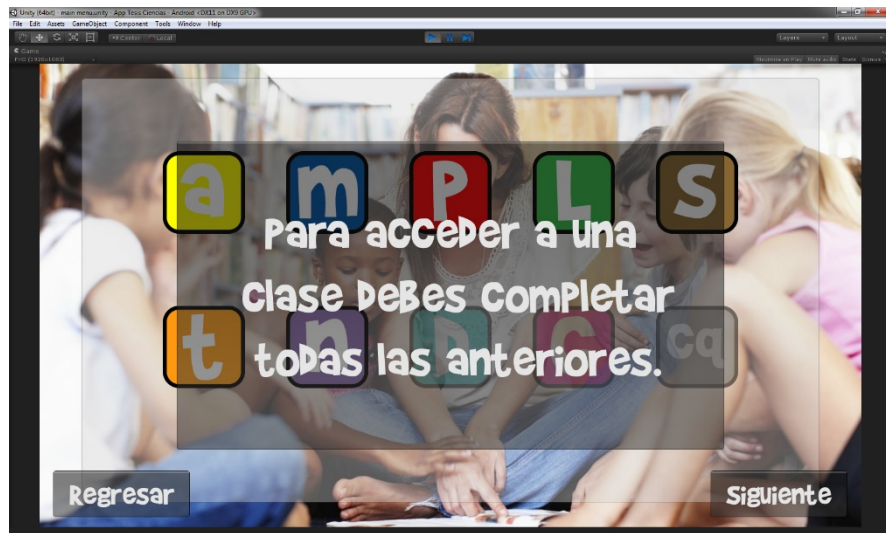


Figura III- 42. Impresion de pantalla de prueba del menú de clases.



Figura III- 43. Impresion de pantalla de prueba de la opción “Reiniciar App”.

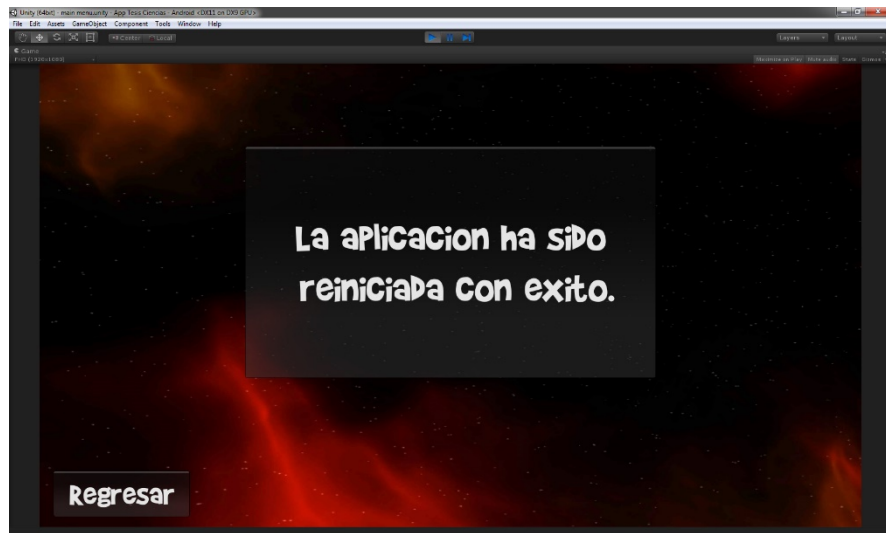


Figura III- 44. Impresion de pantalla de prueba de la función “Reiniciar App”.

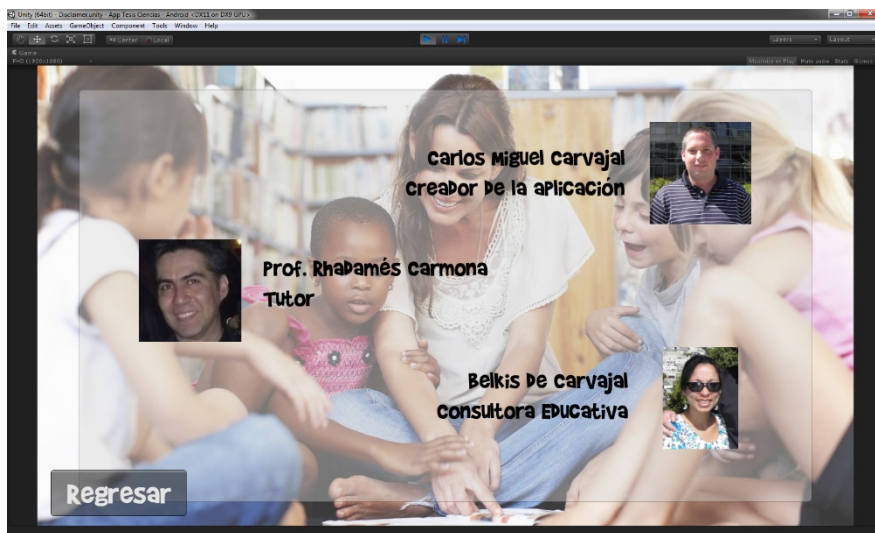


Figura III- 45. Impresion de pantalla de prueba del menú “Créditos”.

3.16 Quinta iteración

A continuación se lista la documentación de la planificación, diseño, codificación y pruebas que tuvieron lugar durante la quinta iteración del proyecto.

3.16.1 Planificación

A continuación se lista la documentación de la planificación de la quinta iteración del proyecto.

3.16.1.1 Historias de usuario

Historia de Usuario	
Identificador: 20	Nombre: Integración de los módulos
Usuario: Usuario	Prioridad: Alta
Programador: Carlos Carvajal	
Descripción: se finaliza la creación de la aplicación integrando todos los módulos anteriores, es decir, clases, mini-juegos, menús, etc. en un solo paquete de aplicación.	

Historia de Usuario	
Identificador: 21	Nombre: Pruebas semifinales.
Usuario: Usuario	Prioridad: Alta
Programador: Carlos Carvajal	
Descripción: se dispone de pruebas semifinales de integración y de todas las funcionalidades de la aplicación como un solo conjunto.	

Historia de Usuario	
Identificador: 22	Nombre: Corrección de errores
Usuario: Usuario	Prioridad: Alta
Programador: Carlos Carvajal	
Descripción: Reunido el equipo de trabajo, se detallaron algunos “bugs” de la aplicación. El programador corrige todos los “bugs” detectados. En esta etapa no solo se evalúa funcionalidades, sino también las características estéticas de la integración.	

Historia de Usuario	
Identificador: 23	Nombre: Prueba final
Usuario: Usuario	Prioridad: Alta
Programador: Carlos Carvajal	
Descripción: se dispone la aplicación para ser probada nuevamente, antes de ser distribuida.	

Historia de Usuario	
Identificador: 24	Nombre: Distribución
Usuario: Usuario	Prioridad: Alta
Programador: Carlos Carvajal	
Descripción: se dispone la aplicación para ser distribuida en <i>App Stores</i> u otros medios.	

3.16.1.2 Plan de entrega

Luego de reunirse todos los actores del proyecto, se determina que para la última entrega se desarrollarían las funcionalidades todas las historias de usuario.

3.16.1.3 Velocidad del proyecto

Se estima un tiempo de desarrollo de nueve (09) días hábiles para la realización de todas las historias de usuario restantes.

3.16.1.4 Tareas específicas

Tarea 20	
Identificador: 20	Número de historia: 20
Nombre: Integración de los módulos.	Tipo: Alta

Inicio: 15/07/2015	Fin: 17/07/2015
Responsable: Carlos Carvajal	
Descripción: se finaliza la creación de la aplicación integrando todos los módulos anteriores, es decir, clases, mini-juegos, menús, etc. en un solo paquete de aplicación.	
Pruebas de aceptación: - Verificar que la aplicación funcione como se espera. - Verificar que el sistema de entrada, audio y video funcione adecuadamente.	

Tarea 21	
Identificador: 21	Número de historia: 21
Nombre: Pruebas semifinales.	Tipo: Alta
Inicio: 20/07/2015	Fin: 21/07/2015
Responsable: Carlos Carvajal	
Descripción: se dispone de pruebas semifinales de integración y de todas las funcionalidades de la aplicación como un solo conjunto.	
Pruebas de aceptación: - Verificar que la aplicación funcione como se espera. - Verificar que el sistema de entrada, audio y video funcione adecuadamente.	

Tarea 22	
Identificador: 22	Número de historia: 22
Nombre: Corrección de errores.	Tipo: Alta
Inicio: 21/07/2015	Fin: 23/07/2015
Programador: Carlos Carvajal	
Descripción: se corrigen todos los “bugs” detectados en la aplicación. Esta etapa no solo evalúa funcionalidades, sino también las características estéticas de la integración.	
Pruebas de aceptación: - Corregir cualquier elemento de la aplicación que no cumpla con los estándares requeridos. - Verificar que la aplicación funcione como se espera.	

- Verificar que el sistema de entrada, audio y video funcione adecuadamente.

Tarea 23	
Identificador: 23	Número de historia: 23
Nombre: Prueba final.	Tipo: Alta
Inicio: 24/07/2015	Fin: 24/07/2015
Programador: Carlos Carvajal	
Descripción: se dispone la aplicación para ser probada nuevamente, antes de ser distribuida.	
Pruebas de aceptación: - Verificar que la aplicación funcione como se espera. - Verificar que el sistema de entrada, audio y video funcione adecuadamente. - Verificar que se cumpla con las evaluaciones de la aplicación por parte de los usuarios (niños).	

Tarea 24	
Identificador: 24	Número de historia: 24
Nombre: Distribución.	Tipo: Alta
Inicio: 27/07/2015	Fin: No disponible.
Programador: Carlos Carvajal	
Descripción: se dispone la aplicación para ser distribuida en <i>App Stores</i> u otros medios.	
Pruebas de aceptación: - Verificar que la aplicación esté disponible al público.	

3.16.1.5 Reuniones

Durante esta primera iteración se llevaron a cabo reuniones cortas y de manera regular, en las que se discutían las tareas a desarrollar, problemas que se podrían presentar, o que se hayan presentado, y sus posibles soluciones. La duración de estas reuniones estuvo comprendida entre quince (15) y treinta (30) minutos.

3.16.2 Diseño

Se procede a integrar todos los módulos de la aplicación, conservando la misma interfaz y los mismos estándares estéticos y funcionales.

3.16.3 Codificación

Toda la codificación durante esta iteración del desarrollo de la aplicación tiene que ver con corrección de errores, relacionados con cambios de valores en ciertas variables, así como optimización de ciertas funciones ya desarrolladas.

3.16. 4 Pruebas

Durante esta etapa se realizaron una serie de pruebas utilizando el editor de *Unity*, basadas en las pruebas de aceptación especificadas en las tareas de desarrollo. Luego se hicieron pruebas en los dispositivos *Android*, y una vez corregidos todos los detalles, se distribuyó a los niños para que fuese probada por ellos ([ver capítulo IV](#) de este trabajo).

Se considera que la aplicación ya esta lista para su distribución al público.

CAPITULO IV:

EVALUACIÓN

Para la realización de esta investigación se tomo como muestra 8 niños participantes, hijos y amigos de los miembros del equipo de pruebas de la aplicación. Estan conformados de la siguiente forma:

DESCRIPCIÓN	SEXO	EDAD (años)
Usuario 1	Masculino	6
Usuario 2	Masculino	5
Usuario 3	Masculino	4
Usuario 4	Masculino	3
Usuario 5	Masculino	3
Usuario 6	Femenino	6
Usuario 7	Femenino	6
Usuario 8	Femenino	5

Tabla N° 3. Descripción del equipo de pruebas de la aplicación. Fuente: Carlos Carvajal 2015.

Como tecnica de recolección de datos se utilizó la observación participante no estructurada, la cual consistió en recoger y anotar todos los hechos que sucedieron en la interacción de los usuarios con los prototipos de la aplicación durante su desarrollo. No se tuvo ninguna guía en particular de lo que se iba a observar.

La intención fue medir el desempeño de los usuarios con respecto al desempeño ideal de lo que el desarrollador espera en los distintos aspectos del manejo de la aplicación.

Como instrumentos de recoleccion de datos se utilizó el cuaderno de notas, el cual es una libreta en la que el observador escribe toda la información, datos,

expresiones, hechos, etc., vistos durante la interacción de cada usuario con la aplicación.

Se realizaron pruebas individualmente a cada usuario. Se siguieron los siguientes pasos:

- Se le entregó un dispositivo móvil según su edad. A los más pequeños (3-4 años) se les entregó el *Samsung Galaxy Note* (GT-N7000), ya que les resultaba menos pesado de cargar. A los más grandes (5-6 años) se les entregó la *tablet Google Nexus 7* (versión 2012).
- Se dieron instrucciones verbales a cada niño(a) acerca de lo que había que hacer en cada actividad.
- Se observó a cada usuario durante un tiempo de 20 minutos, en tres ocasiones a cada uno por separado.
- Cuando el usuario tenía dudas acerca de qué debía hacer se le dieron instrucciones que le ayudaron a continuar.
- Luego de probar la aplicación se le pidió a cada usuario que reconociera un texto en papel que contenía palabras que aprendieron en la aplicación pero con otro diseño y en distinto orden de presentación. Los cuales se muestran a continuación:

MAMÁ

MAMI

MAPA

MAR

PAPÁ

Figura IV-1. Plantilla de evaluación N° 1. Fuente: <http://lagatitacuriosa.blogspot.com/2014/03/analisis-de-un-metodo-de-lectoescritura.html>

nipa

pala

león

alas

papi

SOL

LORO

SOPA

OSO

SAPO

MESA

Figura IV-2. Plantilla de evaluación N° 2.
de evaluación N° 3.

Fuente: [http://lagatitacuriosa.blogspot.com/2014/03/
http://lagatitacuriosa.blogspot.com/2014/03/
analisis-de-un-metodo-de-lectoescritura.html](http://lagatitacuriosa.blogspot.com/2014/03/http://lagatitacuriosa.blogspot.com/2014/03/analisis-de-un-metodo-de-lectoescritura.html)

Figura

IV-3. Plantilla

Fuente:

[analisis-de-un-metodo-de-lectoescritura.html](http://lagatitacuriosa.blogspot.com/2014/03/http://lagatitacuriosa.blogspot.com/2014/03/analisis-de-un-metodo-de-lectoescritura.html)

Resultados

A continuación se resumen algunos aspectos relevantes:

- Los niños afirmaron que el tamaño y peso del dispositivo no es ningún problema.
- Los usuarios preguntaron muy poco acerca de lo que debían hacer, por lo tanto la interacción con el entorno de la aplicación es altamente intuitiva.
- Los botones elegidos para controlar la aplicación fueron unánimemente aceptados.
- Los usuarios reconocían las palabras presentadas en la hoja física que se les entregó. Lo que indica que la aplicación funciona como apoyo en el proceso de aprender a leer.

- Gustaron mucho los juegos intermedios de la aplicación donde el usuario se divertía y distraía de las clases que acababa de aprender.
- En la interacción con el *Circuit Racer* no fue intuitivo el recorrer las alternativas con los botones izquierda derecha arriba abajo, pero una vez que se les explicó cómo se recorrían, les pareció fácil de hacer.
- Los niños valoran mucho los elementos interactivos de la aplicación y la buena calidad de los elementos gráficos
- Cabe destacar que se pudo observar una relación proporcional entre el éxito de la aplicación y la edad del niño, siendo más favorecidos los niños de mayor edad.

CAPÍTULO V:

CONCLUSIONES, RECOMENDACIONES Y TRABAJOS FUTUROS

El presente trabajo especial de grado permitió ampliar los conocimientos del investigador sobre programación en la plataforma *Unity* para dispositivos móviles con fines educativos a nivel de preescolar y primaria.

La aplicación se concibió como una herramienta que pudiera llegar a una cantidad considerable de usuarios. Esto se debe principalmente a que la aplicación fue diseñada y desarrollada para ser ejecutada en dispositivos móviles *Android*. Se realizaron pruebas con dispositivos móviles de modelos relativamente antiguos (*Samsung Galaxy Note* [GT-N7000] y *Google Nexus 7* [versión 2012]). Si bien el desempeño de estos dispositivos para ejecutar la aplicación desarrollada fue más que aceptable, en dispositivos móviles más modernos la experiencia con la aplicación puede mejorar considerablemente, producto de una mejor capacidad de procesamiento, una mayor cantidad de memoria *RAM* para las aplicaciones y una mayor resolución. Si sumamos a esto, el hecho de que los teléfonos móviles que cuentan con los requisitos establecidos para la implementación del software conforman un alto porcentaje del total de los teléfonos móviles en Venezuela, se puede inferir que se asegura la portabilidad de la aplicación a una gran cantidad de dispositivos.

Durante el desarrollo de la aplicación, se realizaron múltiples pruebas de la misma y se evidenció que, aunque la muestra no es representativa, en general, los niños mostraron una muy buena aceptación de la misma, lo cual incrementó notablemente su motivación para usar esta herramienta y aprender a leer. Los resultados de las pruebas revelaron que la aplicación si ayuda y de manera muy significativa, a los niños a aprender a leer.

Finalmente habiendo logrado el diseño y desarrollo de la aplicación móvil así como la recopilación de la información generada a lo largo de este proyecto, se puede afirmar que se cumplieron los objetivos, tanto general como específicos planteados para este trabajo especial de grado.

Algunas recomendaciones para futuras mejoras a la aplicación:

- Subir la aplicación a la tienda *Google Play* para conocer la aceptación de la misma por los usuarios finales mediante sus revisiones y comentarios, así como ampliar la distribución de la misma a nivel nacional/mundial.
- Es necesaria la constante revisión de la aplicación a fin de detectar fallas o posibles mejoras. La constante actualización del sistema operativo *Android*, así como el constante progreso a nivel de *hardware* de dichos dispositivos podrían abrir las puertas a nuevas maneras de interacción, así como a la integración de funcionalidades mejoradas de las ya existentes.
- La mejora de la interfaz gráfica, con ayuda de un diseñador, puede hacer que la aplicación tenga un acabado visual más profesional.

En general se recomienda continuar con investigaciones que promuevan el desarrollo de aplicaciones con fines educativos, ya que los dispositivos móviles poseen grandes ventajas, tales como conectividad, geolocalización, grabación, creación, publicación y el compartir información, entre otros.

Falta por observar detalladamente los efectos que tiene esta aplicación al momento de ser implementada en un salón de clases. Por esto se propone el seguimiento de la misma en uso en una escuela piloto durante un ciclo escolar, para observar el uso que le da el maestro a la aplicación y el aprovechamiento por parte de los niños al utilizarla. Los resultados mostrarían los cambios en el interés hacia el aprender a leer por parte de los niños y el tiempo requerido para que aprendan a hacerlo.

APÉNDICE A: DIAGRAMAS DE CLASES (*GAMEOBJECTS*)

A continuación se presentan todas las clases (*GameObjects*) mencionados en los Diagramas de clases del Capítulo III de este Trabajo.

- A.1 Diagramas de clases para la aplicación en general:

CincoSilabas
<pre>public Font ButtonFont; public string Texto1; public string Texto2; public string Texto3; public string Texto4; public string Texto5; public int TamañoTexto; public AudioClip a; public AudioClip e; public AudioClip i; public AudioClip o; public AudioClip u; public AudioClip muybien; public string ProximaScena; private int clicks1; private int clicks2; private int clicks3; private int clicks4; private int clicks5; private int PlayerLevel; private int MueveTextura1; private int MueveTextura2; private int MueveTextura3; private int MueveTextura4; private int MueveTextura5; public Texture foreground; private int anchoBoton; private int altoBoton; private bool Key; public bool button; public float tapSpeed; private float lastTapTime;</pre>
<pre>void Awake() void Start()</pre>

```

private void MakeSound(AudioClip originalClip)
AudioSource.PlayClipAtPoint(originalClip, transform.position)
public void MakeVoicea()
public void MakeVoiceee()
public void MakeVoiceei()
public void MakeVoiceeo()
public void MakeVoiceeu()
public void MakeVoicemuybien()
private void TestPrintTime1()
private void TestPrintTime2()
private void TestPrintTime3()
private void TestPrintTime4()
private void TestPrintTime5()
IEnumerator ShowMessage ()
public IEnumerator InvokeMethod1(Action method, float interval, int invokeCount)
public IEnumerator InvokeMethod2(Action method, float interval, int invokeCount)
public IEnumerator InvokeMethod3(Action method, float interval, int invokeCount)
public IEnumerator InvokeMethod4(Action method, float interval, int invokeCount)
public IEnumerator InvokeMethod5(Action method, float interval, int invokeCount)
void Update()
IEnumerator delaybien()
void OnGUI()

```

Figura A-1. Clase “Cinco Silabas”. Fuente: Carlos Carvajal 2015.

TocaLaCinco
<pre> public Font ButtonFont; public int TamañoTexto; public string Texto1; public string Texto2; public string Texto3; public string Texto4; public string Texto5; public string Buscado; public AudioClip a; public AudioClip b; public AudioClip c; public AudioClip d; public AudioClip e; public AudioClip buscado; public AudioClip muybien; public AudioClip buscala; public string ProximaScena; public int clickganador; private int click1; </pre>

```

private int click2;
private int click3;
private int click4;
private int click5;
private int MueveTextura1;
private int MueveTextura2;
private int MueveTextura3;
private int MueveTextura4;
private int MueveTextura5;
public Texture foreground;
private string temp;
private int anchoBoton;
private int altoBoton;
public bool button;
public float tapSpeed;
private float lastTapTime;
private bool Key;

void Awake()
void Start()
private void MakeSound(AudioClip originalClip)
AudioSource.PlayClipAtPoint(originalClip, transform.position)
public void MakeVoicea()
public void MakeVoiceb()
public void MakeVoicec()
public void MakeVoiced()
public void MakeVoiceee()
public void MakeVoicebuscala()
public void MakeVoicebuscado()
public void MakeVoicemuybien()
private void TestPrintTime1()
private void TestPrintTime2()
private void TestPrintTime3()
private void TestPrintTime4()
private void TestPrintTime5()
IEnumerator ShowMessage ()
public IEnumerator InvokeMethod1(Action method, float interval, int invokeCount)
public IEnumerator InvokeMethod2(Action method, float interval, int invokeCount)
public IEnumerator InvokeMethod3(Action method, float interval, int invokeCount)
public IEnumerator InvokeMethod4(Action method, float interval, int invokeCount)
public IEnumerator InvokeMethod5(Action method, float interval, int invokeCount)
void Update()
IEnumerator arranca()
IEnumerator delaybien()
IEnumerator delay1()

```

```

IEnumerator delay2()
IEnumerator delay3()
IEnumerator delay4()
IEnumerator delay5()
void OnGUI()

```

Figura A-2. Clase "TocalaCinco". Fuente: Carlos Carvajal 2015.

TresSilabas
<pre> public Font ButtonFont; public string Texto1; public string Texto2; public string Texto3; public int TamañoTexto; public AudioClip a; public AudioClip e; public AudioClip i; public AudioClip muybien; public string ProximaScena; private int clicks1; private int clicks2; private int clicks3; private int PlayerLevel; private int MueveTextura1; private int MueveTextura2; private int MueveTextura3; public Texture foreground; private string temp; private int anchoBoton; private int altoBoton; public bool button; public float tapSpeed; private float lastTapTime; private bool Key; </pre>
<pre> void Awake() void Start() private void MakeSound(AudioClip originalClip) AudioSource.PlayClipAtPoint(originalClip, transform.position) public void MakeVoicea() public void MakeVoicee() public void MakeVoicei() public void MakeVoicemuybien() private void TestPrintTime1() private void TestPrintTime2() private void TestPrintTime3() </pre>

```

IEnumerator ShowMessage ()
public IEnumerator InvokeMethod1(Action method, float interval, int invokeCount)
public IEnumerator InvokeMethod2(Action method, float interval, int invokeCount)
public IEnumerator InvokeMethod3(Action method, float interval, int invokeCount)
void Update()
IEnumerator delaybien()
void OnGUI()

```

Figura A-3. Clase "TresSilabas". Fuente: Carlos Carvajal 2015.

TocaLaTres
<pre> public Font ButtonFont; public int TamañoTexto; public string Texto1; public string Texto2; public string Texto3; public string Buscado; public AudioClip a; public AudioClip b; public AudioClip c; public AudioClip buscado; public AudioClip muybien; public AudioClip buscala; public string ProximaScena; public int clickganador; private int click1; private int click2; private int click3; private int MueveTextura1; private int MueveTextura2; private int MueveTextura3; public Texture foreground; private string temp; private int anchoBoton; private int altoBoton; public bool button; public float tapSpeed; private float lastTapTime; private bool Key; </pre>
<pre> void Start() private void MakeSound(AudioClip originalClip) AudioSource.PlayClipAtPoint(originalClip, transform.position) public void MakeVoicea() public void MakeVoiceb() public void MakeVoicec() </pre>

```

public void MakeVoicebuscala()
public void MakeVoicebuscado()
public void MakeVoicemuybien()
private void TestPrintTime1()
private void TestPrintTime2()
private void TestPrintTime3()
IEnumerator ShowMessage ()
public IEnumerator InvokeMethod1(Action method, float interval, int invokeCount)
public IEnumerator InvokeMethod2(Action method, float interval, int invokeCount)
public IEnumerator InvokeMethod3(Action method, float interval, int invokeCount)
void Update()
IEnumerator arranca()
IEnumerator delaybien()
IEnumerator delay1()
IEnumerator delay2()
IEnumerator delay3()
void OnGUI()

```

Figura A-4. Clase "TocaLaTres". Fuente: Carlos Carvajal 2015.

TresPalabras
<pre> public Font ButtonFont; public string Texto1; public string Texto2; public string Texto3; public int TamañoTexto ; public string ProximaScena; public AudioClip a; public AudioClip b; public AudioClip c; public AudioClip muybien; private int clicks1; private int clicks2; private int clicks3; private int MueveTextura1; private int MueveTextura2; private int MueveTextura3; public Texture foreground; private string temp; private int anchoBoton; private int altoBoton; public bool button; public float tapSpeed; private float lastTapTime; private bool Key; </pre>

```

void Awake()
void Start()
private void MakeSound(AudioClip originalClip)
AudioSource.PlayClipAtPoint(originalClip, transform.position)
public void MakeVoicea()
public void MakeVoiceb()
public void MakeVoicec()
public void MakeVoicemuybien()
private void TestPrintTime1()
private void TestPrintTime2()
private void TestPrintTime3()
IEnumerator ShowMessage ()
public IEnumerator InvokeMethod1(Action method, float interval, int invokeCount)
public IEnumerator InvokeMethod2(Action method, float interval, int invokeCount)
public IEnumerator InvokeMethod3(Action method, float interval, int invokeCount)
IEnumerator Muybien()
IEnumerator delay1()
IEnumerator delay2()
IEnumerator delay3()
void Update()
void OnGUI()

```

Figura A-5. Clase “TresPalabras”. Fuente: Carlos Carvajal 2015.

UnaOracion
<pre> public Texture foreground; public Font ButtonFont; public int TamañoTexto; public string Texto; public string ProximaScena; public AudioClip Voz; public AudioClip muybien; private int clicks; private string temp; private int MueveTextura; private int anchoBoton; private int altoBoton; public bool button; public float tapSpeed; private float lastTapTime; private bool Key; </pre>
<pre> void Start() public void MakeVoicea() public void MakeVoicemuybien() private void MakeSound(AudioClip originalClip) </pre>

```

AudioSource.PlayClipAtPoint(originalClip, transform.position)
IEnumerator ShowMessage ()
IEnumerator Muybien()
IEnumerator delay1()
private void TestPrintTime()
public IEnumerator InvokeMethod(Action method, float interval, int invokeCount)
void Update()
void OnGUI()

```

Figura A-6. Clase "UnaOracion". Fuente: Carlos Carvajal 2015.

MusicLoop
<pre> public int Escena1; public int Escena2; public int Escena3; public int Escena4; public int Escena5; public int Escena6; public int Escena7; public int Escena8; public int Escena9; public int Escena10; public int Escena11; public int Escena12; public int Escena13; public int Escena14; public int Escena15; public int Escena16; public int Escena17; public int Escena18; public int Escena19; public int Escena20; </pre>
<pre> void Start () void Update () DontDestroyOnLoad() </pre>

Figura A-7. Clase "MusicLoop". Fuente: Carlos Carvajal 2015.

- A.2 Diagramas de clases del Mini-juego *Fruit Ninja*

Fruit Spawner
<pre> private GameObject appleReference; private Vector3 throwForce = new Vector3; </pre>
<pre> void Start() </pre>

void SpawnFruit()

Figura A-8. Clase "Fruit Spawner". Fuente: Carlos Carvajal 2015.

Timer
<pre>private GUIText timeTF; public GameObject alertReference; public int nextlevel; public bool button; public float tapSpeed; private float lastTapTime; private bool Key; public Font ButtonFont;</pre>
<pre>IEnumerator ShowMessage () void Start() void Update() void ReduceTime() void OnGUI()</pre>

Figura A-9. Clase "Timer". Fuente: Carlos Carvajal 2015.

Lines Handler
<pre>public Color c1; public Color c2; private GameObject lineGO; private LineRenderer lineRenderer; private int i;</pre>
<pre>void Start() void Update()</pre>

Figura A-10. Clase "Lines Handler". Fuente: Carlos Carvajal 2015.

Splash
<pre>private Color randomAlpha; private float currentAlpha;</pre>
<pre>void Start() void ReduceAlpha()</pre>

Figura A-11. Clase "Splash". Fuente: Carlos Carvajal 2015.

Apple
<pre>private GameObject splashReference; private Vector3 randomPos = new Vector3; private GUIText scoreReference;</pre>
<pre>void Start() void Update() void OnCollisionEnter(Collision other)</pre>

Figura A-12. Clase "Apple". Fuente: Carlos Carvajal 2015.

- A.3 Diagramas de clases del Mini-juego *Roll a Ball*

Player Controller 1
<pre>public float speed; public GUIText PuntosText; public GUIText winText; private int Puntos; private int numberOfGameObjects; public float speedm; public int ProximaEscena; public AudioClip YouWin; public AudioSource MuyBien; private bool hasPlayed; public AudioClip APunto; public AudioSource AudioPunto; private Vector3 zeroAc; private Vector3 curAc; private float sensH; private float sensV; private float smooth; private float.GetAxisH; private float.GetAxisV; private float speedAc;</pre>
<pre>void ResetAxes() void Start() void Update ()</pre>

<pre>void OnTriggerWin() void OnTriggerEnter(Collider other)</pre>
--

Figura A-13. Clase "Player Controller 1". Fuente: Carlos Carvajal 2015.

Camera Follow
<pre>public GameObject targetObject; private float distanceToTarget;</pre>
<pre>void Start () void Update ()</pre>

Figura A-14. Clase "Camera Follow". Fuente: Carlos Carvajal 2015.

Rotator
<pre>void Update ()</pre>

Figura A-15. Clase "Rotator". Fuente: Carlos Carvajal 2015.

- A.4 Diagramas de clases del Mini-juego *Circuit Racer*

Game Controller
<pre>public GameObject myCar; public GameController guiObject; public GUISkin guiSkin; public GameObject background; public Sprite[] backgroundSprites; public float[] startTimesPerLevel; public int[] lapsPerLevel; public int ProximoNivelCar; public AudioClip YouWin; public AudioSource MuyBien; private bool hasPlayed; public Font Fuente; public bool _paused; private float _timeLeft;</pre>

<pre>private float _timePlayed; private int _lapsRemaining; private bool _showingGameOver; private string gameOverText; private float _nextCarAngleTarget; private const float FINISH_TARGET;</pre>
<pre>void Start () void OnTriggerEnter() void PauseGame() void ShowGameOver(bool didWin) void StartNewGame() void OnGUI()</pre>

Figura A-16. Clase "Game Controller". Fuente: Carlos Carvajal 2015.

Car Controller
<pre>public float[] carMaxSpeeds; public AnalogControl analogControl; public Sprite[] carSprites; public AudioClip[] soundEffects; private float _carSpeed; private bool _paused; private bool _stopped;</pre>
<pre>void Start () public void SetPaused(bool isPaused) public void SetCarChoice (int carNum, bool isMultiplayer) public void PlaySoundForLapFinished() public void Stop() void FixedUpdate ()</pre>

Figura A-17. Clase "Car Controller". Fuente: Carlos Carvajal 2015.

Analog Controller
<pre>public GameObject basePad; public GameObject steeringKnob; private Vector2 _direction; private Vector3 _calibration;</pre>

private float _maxRadius; private float _tiltMultiplier; private bool _onMobile; private bool _isUsingPad;
void Start () public Vector2 GetNormalizedSteering()

Figura A-18. Clase "Analog Controller". Fuente: Carlos Carvajal 2015.

BGSound
GameObject[] musicObject;
void Start () void Awake()

Figura A-19. Clase "BGSound". Fuente: Carlos Carvajal 2015.

GUI Controller
public GUIText lapText; public GUIText timeText; private int _laps; private float _timeToShow;
public void SetLaps(int laps) public void SetTime(float timeToShow) void Update ()

Figura A-20. Clase "GUI Controller". Fuente: Carlos Carvajal 2015.

RetainedUserPicksScript
public int carSelected; public int diffSelected; public bool multiplayerGame; private static RetainedUserPicksScript _instance;
public static RetainedUserPicksScript Instance

Figura A-21. Clase "RetainedUserPicksScript". Fuente: Carlos Carvajal 2015.

- A.5 Diagramas de clases del Mini-juego *Angry Letters*

Game Manager
<pre> public Font Fuente; public int ProximaEscenaAngry; public CameraFollowAngry CameraFollowAngry; int currentBirdIndex; public SlingShot slingshot; public static GameState CurrentGameState; private List<GameObject> Bricks; private List<GameObject> Birds; private List<GameObject> Pigs; public AudioClip YouWin; public AudioClip Instructions; public AudioSource MuyBien; private bool hasPlayed; </pre>
<pre> void Start() void Update() private bool AllPigsDestroyed() private void AnimateCameraToStartPosition() void AnimateBirdToSlingshot() private void Slingshot_BirdThrown(object sender, System.EventArgs e) bool BricksBirdsPigsStoppedMoving() public static void AutoResize(int screenWidth, int screenHeight) void OnTriggerEnter() void OnGUI() </pre>

Figura A-22. Clase "Game Manager". Fuente: Carlos Carvajal 2015.

Slingshot
<pre> private Vector3 SlingshotMiddleVector; public SlingshotState slingshotState; public Transform LeftSlingshotOrigin, RightSlingshotOrigin; public LineRenderer SlingshotLineRenderer1; public LineRenderer SlingshotLineRenderer2; public LineRenderer TrajectoryLineRenderer; public GameObject BirdToThrow; public Transform BirdWaitPosition; </pre>

public float ThrowSpeed; public float TimeSinceThrown;
void Start() void Update() private void ThrowBird(float distance) public event EventHandler BirdThrown; private void InitializeBird() void DisplaySlingshotLineRenderers() void SetSlingshotLineRenderersActive(bool active) void SetTrajectoryLineRenderersActive(bool active) void DisplayTrajectoryLineRenderer2(float distance)

Figura A-23. Clase "Slingshot". Fuente: Carlos Carvajal 2015.

Enemy
public float Health; public Sprite SpriteShownWhenHurt; private float ChangeSpriteHealth;
void Start() void OnCollisionEnter2D(Collision2D col)

Figura A-24. Clase "Enemy". Fuente: Carlos Carvajal 2015.

Wood Ball
public BirdState State
void Start() void FixedUpdate() public void OnThrow() IEnumerator DestroyAfter(float seconds)

Figura A-25. Clase "Wood Ball". Fuente: Carlos Carvajal 2015.

Bloque
public float Health = 70f;
void OnCollisionEnter2D(Collision2D col)

Figura A-26. Clase "Bloque". Fuente: Carlos Carvajal 2015.

Camera Controller

public GameObject player;
private Vector3 offset;
void Start ()
void LateUpdate ()

Figura A-27. Clase "Camera Controller". Fuente: Carlos Carvajal 2015.

Camera Fix Aspect Ratio
void Start()

Figura A-28. Clase "Camera Fix Aspect Ratio". Fuente: Carlos Carvajal 2015.

Camera Follow Angry
public Vector3 StartingPosition;
private const float minCameraX;
private const float maxCameraX;
public bool IsFollowing;
public Transform BirdToFollow;
void Start()
void Update()

Figura A-29. Clase "Camera Follow Angry". Fuente: Carlos Carvajal 2015.

Camera Move
private float dragSpeed;
private float timeDragStarted;
private Vector3 previousPosition;
public SlingShot SlingShot;
void Update()

Figura A-30. Clase "Camera Move". Fuente: Carlos Carvajal 2015.

Camera Pinch To Zoom
public float perspectiveZoomSpeed;
public float orthoZoomSpeed;
void Update()

Figura A-31. Clase "Camera Pinch To Zoom". Fuente: Carlos Carvajal 2015.

Constants
namespace Assets.Scripts
public static class Constants

Figura A-32. Clase "Constants". Fuente: Carlos Carvajal 2015.

Destructor
void OnTriggerEnter2D(Collider2D col)

Figura A-33. Clase "Destructor". Fuente: Carlos Carvajal 2015.

Enums
namespace Assets.Scripts
public enum SlingshotState
public enum GameState
public enum BirdState

Figura A-34. Clase "Enums". Fuente: Carlos Carvajal 2015.

- A.6 Diagramas de clases de pertenecientes a la cuarta iteración

Menu Script
private int PlayerLevel; public Font ButtonFont; private string TextoBoton; private int Nivel; private int Expirado; private bool Key; public bool button; public float tapSpeed; private float lastTapTime;
IEnumerator ShowMessage () void Start() void Update() void OnGUI()

Figura A-35. Clase "Menu Script". Fuente: Carlos Carvajal 2015.

Main Menu Buttons 2
public Font ButtonFont;
void OnGUI()

Figura A-36. Clase "Main Menu Buttons 2". Fuente: Carlos Carvajal 2015.

Main Menu Music
public int Escena1; public int Escena2; public int Escena3; public int Escena4; public int Escena5; public int Escena6; public int Escena7; public int Escena8; public int Escena9; public int Escena10; private static MainMenuMusic instance;
public static MainMenuMusic Instance void Awake() void Start () void Update ()

Figura A-37. Clase "Main Menu Music". Fuente: Carlos Carvajal 2015.

Save Game Script
private int PlayerLevel;
void Awake()

Figura A-38. Clase "Save Game Script". Fuente: Carlos Carvajal 2015.

Menu Clases
public int ProximoMenu; public int PrevioMenu; public Font ButtonFont;
void Update() void OnGUI()

Figura A-39. Clase "Menu Clases". Fuente: Carlos Carvajal 2015.

Cargar Escena
public int LevelRequired;
void Update() public void LoadLevel(int levelIndex)

Figura A-40. Clase "Cargar Escena". Fuente: Carlos Carvajal 2015.

Error Clases
private bool MostrarMensaje; public Font Fuente; private int delay;
void Update () public void OnGUI()

Figura A-41. Clase "Error Clases". Fuente: Carlos Carvajal 2015.

App Reset
private bool MostrarMensaje; public Font Fuente; private int delay; public string Mensaje; public string Mensaje1; public string Mensaje2; public InputField Campo; public GameObject Panel; public bool button;
public IEnumerator InvokeMethod(float interval, int invokeCount) IEnumerator Delay() void CheckPassword() void Start () void Update () public void OnGUI()

Figura A-42. Clase "App Reset". Fuente: Carlos Carvajal 2015.

App Reset 2
public int PrevioMenu; public Font ButtonFont;

```
void OnGUI()
```

Figura A-43. Clase "App Reset 2". Fuente: Carlos Carvajal 2015.

Fade IN

```
var fadeTexture : Texture2D;  
var fadeSpeed = 0.2;  
var drawDepth = -1000;  
private var alpha = 1.0;  
private var fadeDir = -1;
```

```
function OnGUI()
```

Figura A-44. Clase "Fade IN". Fuente: Carlos Carvajal 2015.

Fade OUT

```
var fadeTexture : Texture2D;  
var fadeSpeed = 0.2;  
var drawDepth = -1000;  
var waited : boolean = false;  
var NextScene = 0;  
private var alpha = 0.0;  
private var fadeDir = -1;
```

```
function Start()  
function WaitAndPrint (waitTime : float)  
function WaitAndGo (waitTime : float)  
function OnGUI()
```

Figura A-45. Clase "Fade OUT". Fuente: Carlos Carvajal 2015.

APÉNDICE B: MÉTODOS PARA EL DESARROLLO DE SOFTWARE

No sólo con el uso de notaciones de modelado, varias herramientas, disponibilidad de tiempo e interacción grupal se logra desarrollar un software novedoso y exitoso. Es importante considerar que siempre puede haber múltiples aproximaciones y maneras de lograr los objetivos, por lo que para desarrollar un software exitoso también es necesario tomar en cuenta las diferentes metodologías y sus respectivos métodos de desarrollo. Cada metodología tiene una estructura de desarrollo, lo que puede o no repercutir en tiempo, costo, y otros factores durante el desarrollo de software.

En todo método de desarrollo de software se planifica, se asumen roles, se crean actividades, las cuales que pueden ser grupales o no; se hace un modelado; se crea la documentación necesaria, entre otras cosas. La metodología tradicional, a pesar de ser utilizada y ser efectiva en muchos proyectos de gran envergadura, puede no ser la metodología más adecuada para ciertos casos hoy en día, ya que los entornos de los sistemas están en constante cambio. La metodología ligera o ágil tiene diferentes métodos y estrategias que buscan reducir los tiempos de desarrollo manteniendo o incluso mejorando la calidad del producto. Los métodos ágiles están orientados principalmente para desarrollo de proyectos pequeños.

Los métodos ágiles siguen una serie de principios expresados en el “Manifiesto Ágil” [\(3\)](#) publicado en febrero de 2001. Estos principios son:

- El software funcional es la medida principal de progreso.
- La entrega de software funcional debe ser frecuente; semanas en vez de meses.
- Aceptar cambios, incluso en etapas tardías del desarrollo.
- Satisfacer al cliente mediante la entrega temprana y continua de software con valor, es decir, software usable.
- Los responsables del proyecto y los desarrolladores deben trabajar juntos de forma cotidiana durante la realización del mismo.

- Los métodos ágiles promueven el desarrollo sostenible. Los integrantes del grupo de trabajo deben ser capaces de mantener un ritmo constante de forma indefinida.
- El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.
- Los proyectos se desarrollan en torno a individuos motivados. Hay que proveerles el ambiente y el apoyo que necesiten, así como confiarles la ejecución del trabajo.
- La atención continua a la excelencia técnica y al buen diseño mejora la agilidad.
- Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.

B.1 Métodos Ágiles

Ágil se refiere a la capacidad para proveer respuestas rápidas y adaptables al cambio. En el entorno de negocios de hoy en día, estas dos cualidades son indispensables, lo que conlleva a que el software también sea desarrollado de manera ágil.

Las necesidades de un cliente pueden variar desde el momento de inicio de un nuevo proyecto hasta el momento de su entrega final. Esto requiere métodos de desarrollo de software diferentes, que en lugar de rechazar los cambios, sean capaces de adaptarse e incorporarlos. Estos métodos también deben:

- Producir versiones ejecutables en un lapso corto de tiempo con la finalidad de obtener por parte del cliente retroalimentación en cuanto al producto.
- Realizar soluciones sencillas con la finalidad de que el impacto de los cambios que puedan surgir se minimice.

- Ejecutar pruebas continuas y desde temprano, para encontrar y solventar defectos antes que tengan un gran impacto.
- Mejorar la calidad del diseño para así lograr que las iteraciones posteriores requieran menos esfuerzos.

En los últimos años han existido métodos que fomentan las prácticas antes mencionadas, estos son conocidos como “métodos ágiles”.

Según Jim Highsmith y Alistair Cockburn, lo nuevo de estos métodos no son las prácticas que utilizan, sino el reconocimiento de la gente como principal factor de éxito de los proyectos. Adicionalmente, Cockburn define estos métodos como el uso de reglas “ligeras pero suficientes” y la aplicación de técnicas orientadas a las personas y su comunicación [\(20\)](#).

A continuación se detallan algunos de los métodos ágiles más conocidos y aplicados durante el desarrollo de software hoy en día, como lo son: *Adaptive Software Development (Desarrollo Adaptativo de Software, ASD)*, *Agile Unified Process (Proceso Ágil Unificado, AUP)*, *Feature-driven development (Desarrollo Basado en Características, FDD)*, *Extreme Programming (Programación Extrema, XP)*, y *Scrum*.

B.1.1 Adaptive Software Development (Desarrollo Adaptativo de Software, ASD)

Fue desarrollado a comienzos del año 1990 por James Highsmith y Sam Bayer con la intención de que el método se adapte al cambio en lugar de luchar contra él. Se basa en la adaptación continua a circunstancias cambiantes, por lo que es tolerante a los cambios. *ASD* sustituyó al tradicional proceso de diseño secuencial en cascada. En este método no existe un ciclo de planificación, diseño y construcción del software sino más bien un ciclo de especular, colaborar y aprender [\(18\)](#).

B.1.2 Agile Unified Process (Proceso Ágil Unificado, AUP)

Su desarrollo fue propuesto y liderado por Scott Ambler. Es un método de desarrollo de software que está basado en el *Rational Unified Process* (*Proceso Unificado Racional*, RUP) de IBM. El ciclo de vida de AUP es serial en lo grande, e iterativo en lo pequeño. Su objetivo es generar entregables incrementales a lo largo del tiempo⁽⁴⁾. AUP tiene varias disciplinas y fases, las disciplinas son ejecutadas de forma iterativa mientras que las fases son implementadas de forma serial a lo largo del proyecto dentro del cual el equipo de desarrollo define y ejecuta actividades para construir, validar y liberar un software funcional, el cual debe cumplir con las necesidades de los involucrados en su desarrollo.

B.1.3 Feature-driven development (Desarrollo Orientado a las Características, FDD)

Es un método ágil iterativo e incremental, adaptativo y orientado por características (*features*). A diferencia de otros métodos ágiles no cubre todo el ciclo de vida sino sólo las fases de diseño y construcción⁽⁴²⁾. Se complementa con otros métodos ya que no precisa de un modelo específico. Se enfoca en calidad, entregas funcionales constantes y formas de evaluación del proyecto. Su ciclo de vida está compuesto por cinco procesos: desarrollo del modelo general, construcción de la lista de características, planificación por características, diseño por características y construcción por características. Se rige por los siguientes principios:

- Es mejor trabajar un proceso simple y bien definido.
- Los pasos de un proceso deben ser lógicos, vanagloriarse del proceso puede impedir el trabajo real.

B.1.4 Extreme Programming (Programación Extrema, XP)

Fue desarrollado en el año 1999 por Kent Beck ⁽²⁾ y es uno de los métodos ágiles de desarrollo de software más destacados. Se diferencia de los métodos tradicionales en su enfoque hacia la adaptabilidad y previsión ante los cambios. Se considera que los cambios suelen ser frecuentes y aceptados durante todo el desarrollo del proyecto ya que por naturaleza es muy difícil especificar todos los requisitos al comienzo del mismo.

Extreme Programming es un método de desarrollo incremental iterativo que busca reducir la complejidad del proceso de desarrollo de software mediante un trabajo orientado a la obtención inmediata de objetivos ⁽²⁾.

Se podría considerar *XP* como uno de los mejores métodos para el desarrollo de proyectos, ya que aparte de lo anterior, está centrado en potenciar las relaciones interpersonales promoviendo el trabajo en equipo y el éxito del mismo, preocupándose por el aprendizaje del personal y proporcionando un entorno cómodo para el trabajo.

XP se basa principalmente en la interacción entre el cliente y el equipo de desarrollo, simplicidad en las soluciones, comunicación fluida y constante entre equipos de trabajo, y coraje para enfrentar cambios. *XP* suele ser muy útil para proyectos con requisitos imprecisos y muy variantes.

Con *XP* se realizan mejoras constantemente ya que el desarrollo es iterativo e incremental, también se hacen pruebas unitarias, continuas, repetitivas y frecuentemente automatizadas.

B.1.5 Scrum

Es un proceso iterativo e incremental para la administración de proyectos enfocado en la administración de proyectos de desarrollo de *software* ⁽³⁷⁾.

Durante cada sprint (ciclo del proceso que dura por lo general entre dos y cuatro semanas), el equipo de desarrollo crea un producto probado y funcional.

Durante este período se mantiene un *sprint backlog*, el cual contiene un conjunto de tareas de alta prioridad a realizarse, las cuales son determinadas en una reunión previa a cada sprint. Durante esta reunión el cliente le indica al equipo de desarrollo qué tareas deberían incluirse en el *sprint backlog*. Para el próximo sprint, el equipo analiza la viabilidad (tiempo, carga de trabajo, etc.) de la petición y agrega la tarea al *sprint backlog*, concretando así los requerimientos del mismo.

Todos los días durante el *sprint* se lleva a cabo una reunión de 15 minutos en promedio, en la cual se discuten puntos de planificación, objetivos logrados y objetivos por cumplir.

B.2 Características de *Extreme Programming (XP)*

Antes de comenzar un proyecto *XP* se deben analizar cuatro variables: costo, tiempo, calidad y ámbito del proyecto. *XP* propone hacer un análisis en donde se pondere la importancia de cada variable, debido al impacto que cada una puede ejercer sobre las otras. *XP* se basa en la mejora continua del código fuente mediante prácticas de comunicación entre los actores participantes y reutilización de artefactos. Se busca mantener una alta calidad en los artefactos producidos mientras estos se generen rápidamente. El proceso de desarrollo *XP* es iterativo e incremental, sin introducir nuevas funcionalidades al sistema antes que estas sean necesarias. El proceso está orientado a los desarrolladores y al usuario, los cuales interactúan entre sí para definir el alcance del proyecto y corregir errores conceptuales a la brevedad posible.

B.3 Actores en *Extreme Programming (XP)*

XP define cierto conjunto de actores que intervienen durante el proceso de desarrollo, cada uno con responsabilidades específicas:

- Programadores: son los actores responsables de la construcción del sistema, tomando las decisiones técnicas necesarias. Son quienes programan y diseñan. En *XP* no se realiza distinción entre analista y programador.
- Jefe de Proyecto: es el encargado de organizar y guiar las reuniones del grupo de trabajo, además asegura las condiciones de trabajo adecuadas para el desarrollo del proyecto.
- Cliente: en *XP*, se considera al cliente como parte del equipo de desarrollo ya que determina qué y cuándo se va a construir, además de establecer las pruebas funcionales.
- Encargado de Pruebas: en muchas ocasiones el cliente no posee conocimientos computacionales, por lo que un encargado de pruebas debe asesorarlo para la definición y cumplimiento de pruebas funcionales.

B.4 Fases del Extreme Programming (XP)

El proceso de desarrollo está compuesto por las etapas de planificación de proyecto, diseño, implementación y pruebas, las cuales aparecen en la siguiente figura:

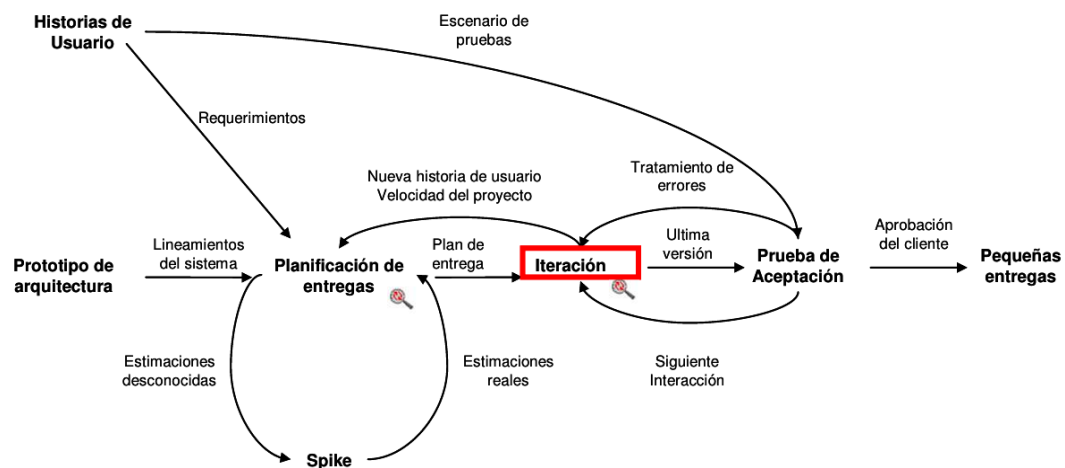


Figura B.1 - Fases de Desarrollo de la Programación Extrema. Fuente:

<http://oness.sourceforge.net/proyecto/html/ch05s02.html>

B.5 Planificación del Proyecto

Al iniciar un proyecto en *XP* se definen historias de usuario, las cuales capturan los requerimientos del mismo. Estas historias son breves, constan de 3 o 4 líneas escritas por el cliente, mencionando sin detalles los requerimientos de una porción del sistema. Idealmente, cada historia de usuario puede ser desarrollada en un plazo de 1 a 3 semanas.

El siguiente paso en *XP* es la planificación de entregas, cada una de ellas debe abarcar un conjunto de funcionalidades relacionadas con las historias de usuario. Al especificar las historias de usuario se deben tener en cuenta las prioridades y dependencias entre ellas, además de su valor para el sistema o el cliente. En base a esta planificación se establecen fechas de entrega, para dejar así establecidos los tiempos de desarrollo. En el plan se incluye el grupo de trabajo asignado y las pruebas a realizar para verificar la calidad del producto.

Una vez establecido un plan de entregas, se divide el proyecto en iteraciones cuya duración debe oscilar entre una y tres semanas. Al comienzo de cada entrega, el cliente selecciona las historias de usuario definidas en el plan de entrega que deben ser implementadas, lo cual incluye las historias de usuario que no pasaron la prueba de aceptación en la iteración anterior.

B.6 Diseño de la Aplicación

La aplicación se diseña utilizando tarjetas de Historias de Usuarios donde cada una representa una clase con responsabilidades definidas, así como un listado de clases con las que debe colaborar. A continuación se muestra la figura B-2 como ejemplo:

Historia de Usuario	
Identificador:	Nombre:
Usuario:	Prioridad:
Programador:	
Descripción:	

Figura B.2- Estructura de una Tarjeta de Historia de Usuario. Fuente: <http://ingsoftware072301.obolog.es/metodologia-xp-2012877>

Estas tarjetas tienen la intención de simplificar el diseño orientado a objetos, distribuyendo las tareas entre el grupo y evitando la duplicación de responsabilidades.

XP sugiere el uso de diseños simples de fácil implementación, que proporcionen sólo las funcionalidades necesarias. También sugiere que los nombres usados por las clases y métodos usen un mismo glosario de términos acordado por el grupo de desarrollo.

XP promueve el uso de código existente, con la condición que sea refactorizado antes de ser incluido en el proyecto. Aquí refactorizar se refiere a modificar la estructura del código fuente existente sin alterar sus funcionalidades. El código refactorizado debe usar el glosario de nombres determinado por el grupo de desarrollo, no usar diseños obsoletos y sólo contener las funcionalidades que serán necesarias en el proyecto.

B.7 Implementación de la Aplicación

Luego de determinar cuáles son las historias de usuario que se implementarán en la iteración actual, se pide al cliente que especifique con detalle lo que debe implementarse. Se sugiere el uso de estándares existentes durante la implementación, ya que mantiene el código consistente y facilita la comprensión, escalabilidad y mantenimiento de la aplicación.

Para no desestabilizar el código fuente existente, se sugiere el uso de software de control de versiones, como *Git* y/o *Mercurial*. Un beneficio del uso de control de

versiones es que permite al grupo de desarrollo trabajar simultáneamente sobre el código fuente sin que algún miembro use por error una versión antigua.

Cuando se desarrolla usando *Extreme Programming*, existe un énfasis fuerte en que el código fuente sea claro y fácil de entender, para que cualquier programador del grupo de trabajo pueda modificar y extender el código existente. Esto se hace con la intención de repartir equitativamente las tareas en el grupo y no hacer a alguna persona particularmente indispensable.

B.8 Pruebas a la Aplicación

La calidad del producto final es garantizada a partir de pruebas de unidad (*unit tests*), que son realizadas a medida que se va desarrollando el producto. Estas pruebas deben ser automáticamente verificadas, ya que el desarrollador conoce el resultado que debe arrojar la ejecución de la prueba.

Las pruebas de unidad deben ser almacenadas junto con el código fuente probado en el repositorio de código, de forma que puedan ser modificadas a medida que se añadan funcionalidades al componente a probar.

Las pruebas de unidad deben ser independientes a la implementación del componente, deben utilizar la interfaz proporcionada por el componente a probar. Esto es ventajoso para el desarrollo porque:

- Mejora el diseño de la aplicación, ya que éste debe hacerse orientado al componente y su correspondiente prueba.
- Facilita la refactorización de código y la ejecución de las pruebas de unidad verifica rápidamente que un cambio realizado sea correcto

B.9 Buenas Prácticas en XP

Para conseguir resultados de calidad en un tiempo y costo razonable, *XP* sugiere aplicar los siguientes principios.

- Planificación del proyecto: se sugiere construir un plan inicial para el desarrollo del sistema y refinarlo a medida que avanza, de forma que se ajuste a los eventos que ocurran durante el desarrollo.
- Diseño simple: *XP* establece que sólo deben añadirse funcionalidades que forme parte de los requerimientos analizados, esto se hace para minimizar los errores y acelerar el proceso de desarrollo.
- Cliente en el lugar: debe contarse con el cliente durante todo el proceso de planificación y desarrollo del proyecto, a fin de aclarar dudas y tomar decisiones importantes, esto reduce los problemas y malentendidos que retrasen el desarrollo.
- Entregas pequeñas: implica la planificación y entrega de versiones funcionales frecuentemente, para que el cliente evalúe y retroalimente el proceso de desarrollo.
- Estándares de código: el código debe seguir una estructura similar en todo el proyecto, para facilitar la refactorización, construcción de pruebas de unidad y aligerar la labor de comprensión del código por parte de otros desarrolladores.
- Pruebas del sistema: se deben realizar pruebas de unidad y pruebas de aceptación. Las pruebas de unidad deben ser creadas por los desarrolladores que producen el código fuente y el cliente debe crear las pruebas de aceptación, las cuales evalúan el cumplimiento de los requerimientos del sistema.

- Programación en equipos de dos: se sugiere la producción de código por equipos de dos desarrolladores compartiendo una misma computadora. Esta práctica tiene como intención lograr la producción de código de mayor calidad.
- Propiedad colectiva del código: todos los desarrolladores deben ser capaces de realizar cambios en el código fuente. La implementación de los módulos del sistema será distribuida entre todos para lograr una capacitación colectiva en todo el código del proyecto y así conseguir flexibilidad a la hora de distribuir tareas.
- Integración continua: se sugiere integrar el código frecuentemente para evitar los errores que surgen al integrar módulos que deben interactuar.
- Cuarenta horas de trabajo a la semana: *XP* especifica que cada desarrollador debe trabajar 8 horas diarias por 5 días a la semana, lo cual permite el aprovechamiento del recurso humano sin reducir la calidad del producto final.

APÉNDICE C: ANDROID DEVELOPER TOOLS Y UNITY

A continuación se presentan las principales herramientas que se utilizarán para el desarrollo de esta aplicación para dispositivos *Android*.

C.1 Android Developer Tools (Herramientas para Desarrolladores en Android, ADT):

ADT (Android Developer Tools) es un *plug-in* para *Eclipse* que proporciona una suite de herramientas integradas al *IDE Eclipse*. Ofrece acceso gráfico (*GUI*) a muchos de los comandos en línea de las herramientas del *SDK* así como una herramienta de diseño de interfaz de usuario (*UI*) para elaboración rápida de prototipos, de diseños y la construcción de interfaces de usuario de aplicaciones.

Debido a que *ADT* es un *plug-in* para *Eclipse*, se obtiene la funcionalidad de un *IDE* bien establecido, junto al acceso a características específicas para *Android*, las cuales proporciona el *ADT*. Las siguientes son las características más importantes de *Eclipse* y *ADT*:

- Creación integrada, construcción, empaquetado, instalación y depuración de proyectos *Android*: *ADT* integra en *Eclipse* muchas de las tareas del flujo del desarrollo de software, agilizando el desarrollo y las pruebas de las aplicaciones *Android*.
- Integración de herramientas del *SDK*: muchas de las herramientas del *SDK* están integradas en los menús de *Eclipse*, perspectivas o como parte de los procesos de fondo ejecutados por *ADT*.
- Leguaje de programación *Java* y editores *XML*: el lenguaje de programación *Java* contiene características *IDE* comunes, tales como la revisión de sintaxis en tiempo de compilación, auto-completar, y una documentación integrada para los *frameworks* de las *APIs Android*.

ADT también provee editores *XML* personalizados que permiten editar archivos *XML* específicos para *Android*, con una interfaz tipo formulario. Un editor gráfico permite diseñar interfaces con comportamiento “seleccionar y soltar” (drag and drop).

- Documentación integrada para los *frameworks* integrados *APIs*: se puede ver documentación con simplemente pasar el mouse sobre clases, métodos o variables.

El desarrollo de software *Android* es el proceso mediante el cual nuevas aplicaciones son creadas para el sistema operativo *Android*. Éstas aplicaciones generalmente son creadas bajo el lenguaje de programación *Java*, utilizando en Kit de Desarrollo de Software (*SDK*) *Android*; aunque existen otras herramientas de desarrollo.

Para Julio del 2013, más de un millón de aplicaciones *Android* han sido desarrolladas, con más de 25.000 millones de descargas en total. Para Junio del 2011, investigaciones muestran que más del 67% de los desarrolladores de software móvil han utilizado esta plataforma. En el segundo trimestre del 2012, cerca de 105 millones de dispositivos móviles *Android* fueron distribuidos, con lo cual se logró una porción de mercado del 68%, entre todos los dispositivos móviles distribuidos.

***Android* SDK:**

El Kit de Desarrollo de *Software* (*SDK*) *Android* incluye una completa lista de herramientas de desarrollo. Entre ellas están un depurador, librerías, un emulador de dispositivo, documentación, código de ejemplo y tutoriales. Entre las plataformas actualmente soportadas se incluyen computadoras corriendo *Linux* (cualquier distribución moderna), *Mac OS X* 10.5.8 o superior y *Windows XP* o superior. El ambiente de desarrollo integrado (*IDE*) oficialmente soportado es *Eclipse*, usando el *plug-in ADT*; a pesar de que el *IDE IntelliJ IDEA* soporta completamente el desarrollo *Android*, así como también *Netbeans IDE*, que soporta el desarrollo *Android* a través de un *plug-in*. Adicionalmente, los desarrolladores pueden utilizar cualquier editor de

texto para editar archivos *Java* y *XML*, y utilizar las herramientas por línea de comandos (*Java Development Kits* y *Apache Ant* son requeridos) para crear, compilar y depurar aplicaciones *Android*, así como dispositivos de control *Android*.

Las mejoras del *SDK Android* van mano a mano con la plataforma de desarrollo *Android*. El *SDK* también soporta versiones antiguas de la plataforma *Android* en caso de que los desarrolladores deseen utilizar sus aplicaciones en equipos más antiguos. Las herramientas de desarrollo son componentes descargables, de manera que al haber descargado la versión más reciente, versiones más antiguas pueden ser descargadas también, algo muy útil para probar compatibilidad.

Las aplicaciones *Android* son empacadas en formato *.apk* y almacenadas bajo la carpeta */data/app* en el sistema operativo *Android* (esta carpeta solo es accesible por root, debido a razones de seguridad). Los paquetes *APK* contienen archivos *.dex* (archivos de código binario compilados, llamados ejecutables *Dalvik*), archivos de recursos, etc.

C.2 Unity

Unity es un motor de videojuego multiplataforma creado por *Unity Technologies*. *Unity* está disponible como plataforma de desarrollo para Microsoft Windows y OS X, y permite crear juegos para *Windows*, *OS X*, *Linux*, *Xbox 360*, *PlayStation 3*, *Playstation Vita*, *Wii*, *Wii U*, *iPad*, *iPhone*, *Android* y *Windows Phone*. Gracias al *plug-in* web de *Unity*, también se pueden desarrollar videojuegos de navegador para *Windows* y *Mac*.

Unity Technologies

La empresa *Unity Technologies* fue fundada en 2004 por David Helgason (CEO), Nicholas Francis (CCO), y Joachim Ante (CTO) en Copenhague, Dinamarca después de su primer juego, *GooBall*, que no obtuvo éxito. Los tres reconocieron el valor del motor y las herramientas de desarrollo y se dispuso a crear un motor que se

pudiera utilizar a un precio asequible. Para tal fin, *Unity Technologies* ha recibido financiación de la talla de Sequoia Capital, Capital WestSummit y Socios iGlobe.

El éxito de *Unity* ha llegado en parte debido al enfoque en las necesidades de los desarrolladores independientes que no pueden crear ni su propio motor del juego, ni las herramientas necesarias o adquirir licencias para utilizar plenamente otras opciones disponibles. El enfoque de la compañía es "democratizar el desarrollo de juegos", y hacer el desarrollo de contenidos interactivos en 2D y 3D lo más accesible posible.

En 2008, con el auge del *iPhone*, *Unity* fue uno de los primeros desarrolladores de motores en empezar a apoyar a la plataforma en su totalidad. *Unity* está siendo utilizado por el 53,1% de los desarrolladores (según la encuesta de *Game Developer* tecnología móvil y social), con cientos de juegos lanzados en dispositivos *Android* e *iOS*.

En 2009, *Unity* comenzó a ofrecer una versión de su producto de forma gratuita. El número de desarrolladores registrados empezó a crecer rápidamente tras el anuncio. En abril de 2012, *Unity* se dio cuenta del nuevo nivel popularidad a medida que el recuento de los desarrolladores registrados alcanzaron 1 millón, 300.000 de los cuales utilizan *Unity* constantemente.

Historia de *Unity*

La primera versión de *Unity* se lanzó en la Conferencia Mundial de Desarrolladores de Apple en 2005. Fue construido solamente para funcionar y generar proyectos en los equipos de la plataforma *Mac* y obtuvo el éxito suficiente como para continuar con el desarrollo del motor y las herramientas. *Unity 3* fue lanzado en septiembre de 2010 y se centró en empezar a introducir más herramientas que los estudios de alta gama por lo general tienen a su disposición, con el fin de captar el interés de los desarrolladores más grandes, mientras que proporciona herramientas para equipos independientes y más pequeñas que normalmente serían difíciles de conseguir en un paquete asequible. La última versión de *Unity*, *Unity 4*,

lanzada a finales de 2012, se anunció en junio de 2012 e incluye añadidos como *Mecanim animation*, soporte para *DirectX 11* y soporte para juegos en *Linux*.

C.2.1 Características de *Unity*

Unity puede usarse junto con 3d *Studio Max*, *Maya*, *Softimage*, *Blender*, *Modo*, *ZBrush*, *Cinema 4D*, *Cheetah3D*, *Adobe Photoshop*, *Adobe Fireworks* y *Allegorithmic Substance*. Los cambios realizados a los objetos creados con estos productos se actualizan automáticamente en todas las instancias de ese objeto durante todo el proyecto sin necesidad de volver a importar manualmente.

El motor gráfico utiliza *Direct3D* (en *Windows*), *OpenGL* (en *Mac* y *Linux*), *OpenGL ES* (en *Android* e *iOS*), e interfaces propietarias (*Wii*). Tiene soporte para mapeado de relieve, reflexión de mapeado, mapeado por paralaje (*parallax mapping*), pantalla de espacio oclusión ambiental (*SSAO*), sombras dinámicas utilizando mapas de sombras, render a textura y efectos de post-procesamiento de pantalla completa.

Se usa *ShaderLab language* para el uso de *shaders*, soporta tanto programación declarativa de los programas de función fija de tuberías y *shader GLSL* o escritas en *Cg*. Un *shader* puede incluir múltiples variantes y una especificación declarativa de reserva, lo que permite a *Unity* detectar la mejor variante para la tarjeta de vídeo actual y si no son compatibles, recurrir a un *shader* alternativo que puede sacrificar características para una mayor compatibilidad.

El soporte integrado para *Nvidia* (antes *Ageia*), el motor de física *PhysX*, (a partir de *Unity 3.0*) con soporte en tiempo real para mallas arbitrarias y sin piel, *ray casts* gruesos, y las capas de colisión.

El *scripting* viene a través de *Mono*. El *script* se basa en *Mono*, la implementación de código abierto de *.NET Framework*. Los programadores pueden utilizar *UnityScript* (un lenguaje personalizado inspirado en la sintaxis *ECMAScript*),

C# o *Boo* (que tiene una sintaxis inspirada en *Python*). A partir de la versión 3.0 añade una versión personalizada de *MonoDevelop* para la depuración de *scripts*.

Unity también incluye *Unity Asset Server* - una solución de control de versiones para todos los *assets* de juego y *scripts*, utilizando *PostgreSQL* como *backend*, un sistema de audio construido con la biblioteca *FMOD*, con capacidad para reproducir audio comprimido *Ogg Vorbis*, reproducción de vídeo con códec *Theora*, un motor de terreno y vegetación , con árboles con soporte de *billboarding*, determinación de cara oculta con *Umbral*, una función de iluminación *lightmapping* y global con *Beast*, redes multijugador *RakNet* y una función de búsqueda de caminos en mallas de navegación.

Unity soporta un montón de nuevas características potentes. Estas incluyen el sistema de partículas *Shuriken*, *navmesh* para *pathfinding* y evasión de obstáculos, iluminación del espacio lineal (gamma correcta), renderización HDR, renderización multihilo, sondas de luz, despliegue *Google Native Client*, reescrito desecho oclusión, una función de nivel de detalle apoyo, un *add-on* de *Adobe Flash Player* para vista previa, perfilador de *GPU*, y *lightmaps* direccionales.

Mecanim

Mecanim es la tecnología de animación de *Unity*, que ha estado en desarrollo durante años, primero por la empresa del mismo nombre, a continuación, por las oficinas de *Unity* en Canadá, después de su adquisición. La tecnología está diseñada para llevar el movimiento fluido y natural de los personajes con una interfaz eficiente. *Mecanim* incluye herramientas para la creación de máquinas de estados, árboles de mezcla, manipulación de los conocimientos indígenas y *retargeting* automático de animaciones, desde el editor de *Unity*.

Además, una serie de animaciones redestinables estarán disponible en el *Unity Asset Store de Unity* tras el lanzamiento de la herramienta. Muchos de estos archivos de animación de captura de movimiento son proporcionados sin costo por

Unity Technologies. Otros proveedores del *Asset Store* también ofrecerán animaciones para su uso con *Mecanim*, ya sea de forma gratuita o de pago.

DirectX 11

Unity 4 es compatible con *Microsoft DirectX 11*.

Mejores gráficos móviles

Unity 4 cuenta con sombras en tiempo real en móviles, creación de instancias, malla de piel, la capacidad de utilizar los mapas normales cuando se usan mapas de *lightmaps*, y un refinado perfilador de *GPU*. Es más fácil hacer gráficos de gama alta escalables a los modernos *PCs* y chips gráficos móviles.

Unity add-on para Adobe Flash Player

El despliegue de *Adobe Flash Player add-on* también empieza con *Unity 4.0*. Mientras que el despliegue ha sido posible para las personas con *Unity beta 3.5* compatible, la versión final de la implementación del *add-on* requiere *Unity 4*.

Publicación en Linux

Unity también incluye una nueva opción de implementación para publicar juegos en el escritorio de *Linux*. Si bien la implementación del *add-on* puede trabajar con diversas formas de *Linux*, el desarrollo se centra principalmente en Ubuntu en su liberación primaria. Esta opción de despliegue se proporcionará a todos los usuarios de *Unity 4*, sin costo adicional. Los ingenieros de *Unity* trabajaron con el equipo de Ubuntu Canonical para los juegos.

La plataforma del *Unity* entra en su cuenta como una herramienta de animación de gran alcance que está totalmente integrado con el motor de la unidad, este incluye el nuevo ajuste "*Dopesheet*", basada en fotogramas clave en la ventana

de animación para animar *sprites* o "*Mecanim*" para impulsar forma de mezclas de animaciones faciales.

Plataforma “crear una vez, implementar en todas partes”

Unity soporta el despliegue de múltiples plataformas. *Unity* le permite orientar todas las plataformas y cambiar entre ellas desde una sola herramienta. Dentro de un proyecto que tiene control sobre la implementación a todas las plataformas, incluyendo teléfonos móviles, webs, escritorios y consolas. *Unity* hace que sea fácil de mantener el código de trabajo a través de muchos dispositivos de abstracción de la mayoría de las diferencias de la plataforma, manteniendo la opción para tener un control preciso cuando sea necesario. *Unity* también permite la especificación de compresión de texturas y ajustes de resolución para cada plataforma para la que un juego es compatible. Esto significa que un solo archivo de alta resolución funcionará para todos los destinos. Al tomar el duro camino del proceso de desarrollo y hacer todo el trabajo de fondo, *Unity* permite a los desarrolladores centrarse en hacer juegos.

Licencias

Hay dos licencias principales para desarrolladores: *Unity* y *Unity Pro*, que está disponible por un precio ya que la versión *Pro* no es gratis. Originalmente costaba alrededor de 200 dólares estadounidenses. La versión *Pro* tiene características adicionales, tales como *render* a textura, determinación de cara oculta, iluminación global y efectos de post-procesamiento. La versión gratuita, por otro lado, muestra una pantalla de bienvenida (en juegos independientes) y una marca de agua (en los juegos web) que no se puede personalizar o desactivar.

Tanto *Unity* como *Unity Pro* incluyen el entorno de desarrollo, tutoriales, ejemplos de proyectos y de contenido, soporte a través de foros, *wiki*, y las actualizaciones futuras de la misma versión principal (es decir, la compra *Unity Pro 4* obtiene todas las futuras actualizaciones de *Unity Pro 4.x* gratis).

Unity para *Android*, *Unity* para *iOS*, *Unity* para *Adobe Flash Player*, y *Unity* para teléfonos con *Windows 8* son complementos para una compra de *Unity*. Es obligatorio el certificado de *Unity Pro* para comprar licencias Pro para *Android* o *iOS*. Las licencias normales de *Android* e *iOS* se pueden utilizar con la versión gratuita de *Unity*.

El código fuente, *PlayStation 3*, *Xbox 360*, *Wii* y licencias se negocian caso por caso.

Las licencias educativas son proporcionadas por *Studica* con la estipulación de que es para la compra y uso de las escuelas, exclusivamente para la educación.

Desde la versión 4.0, un nuevo modelo de licencia se puso en marcha para organizaciones de juegos de azar. Deben ponerse en contacto con *Unity* directamente para obtener una licencia de distribución. Esta licencia se encuentra en el nivel de la distribución, no el nivel de desarrollador.

Iniciativas de democratización

Para reforzar su democratización en el desarrollo del juego, *Unity Technologies* invierte en iniciativas que las considera como vías para ayudar a capacitar a los desarrolladores mediante la ampliación de sus capacidades y el alcance del cliente.

Asset Store

En noviembre de 2010 se lanzó el *Unity Asset Store* que es un recurso disponible en el editor de *Unity*. Más de 150.000 usuarios de *Unity* pueden acceder a la colección de más de 4.400 paquetes de *Assets* en una amplia gama de categorías, incluyendo modelos 3D, texturas y materiales, sistemas de partículas, música y efectos de sonido, tutoriales y proyectos, paquetes de *scripts*, extensiones para el editor y servicios en línea.

Union

Union19 es una división de *Unity Technologies* dedicada a la sindicación de los juegos de *Unity* para teléfonos móviles, tiendas de aplicaciones, tabletas, decodificadores, televisores conectados y otras plataformas emergentes. Con el objetivo de democratizar la distribución de juegos, *Union* trabaja con desarrolladores de *Unity* en licencias de juegos para el lanzamiento en los nuevos dispositivos.

Union incluye una gama de más de 125 juegos que ha generado un total acumulado de 120 millones de descargas en sus comunicados colectivos. Ejemplos de títulos de Unión incluyen *Shadowgun*, *Super Crossfire HD*, *Forever Frisbee*, *Falling Fred*, y *Cordy*.

Union ofrece a los socios de la plataforma acceder a los juegos mientras fortalecen a los desarrolladores de *Unity* con nuevas oportunidades de distribución. Union es libre de unirse y proporciona el 80% de cuota de ingresos de sus desarrolladores.

Las plataformas de *Union* incluyen *Intel*, *LGTV*, *Roku*, *BlackBerry*, *Nokia*, *Sony* y *Lenovo*.

APÉNDICE D: MARCO LEGAL

D.1 CONSTITUCIÓN DE LA REPÚBLICA BOLIVARIANA DE VENEZUELA

TÍTULO III - De los Derechos Humanos y Garantías, Y de los Deberes

Capítulo VI. De los Derechos Culturales y Educativos

Artículo 98. La creación cultural es libre. Esta libertad comprende el derecho a la inversión, producción y divulgación de la obra creativa, científica, tecnológica y humanística, incluyendo la protección legal de los derechos del autor o de la autora sobre sus obras. El Estado reconocerá y protegerá la propiedad intelectual sobre las obras científicas, literarias y artísticas, invenciones, innovaciones, denominaciones, patentes, marcas y lemas de acuerdo con las condiciones y excepciones que establezcan la ley y los tratados internacionales suscritos y ratificados por la República en esta materia.

Artículo 102. La educación es un derecho humano y un deber social fundamental, es democrática, gratuita y obligatoria. El Estado la asumirá como función indeclinable y de máximo interés en todos sus niveles y modalidades, y como instrumento del conocimiento científico, humanístico y tecnológico al servicio de la sociedad. La educación es un servicio público y está fundamentada en el respeto a todas las corrientes del pensamiento, con la finalidad de desarrollar el potencial creativo de cada ser humano y el pleno ejercicio de su personalidad en una sociedad democrática basada en la valoración ética del trabajo y en la participación activa, consciente y solidaria en los procesos de transformación social consustanciados con los valores de la identidad nacional, y con una visión latinoamericana y universal. El Estado, con la participación de las familias y la sociedad, promoverá el proceso de educación ciudadana de acuerdo con los principios contenidos de esta Constitución y en la ley.

Artículo 110. El Estado reconocerá el interés público de la ciencia, la tecnología, el conocimiento, la innovación y sus aplicaciones y los servicios de

información necesarios por ser instrumentos fundamentales para el desarrollo económico, social y político del país, así como para la seguridad y soberanía nacional. Para el fomento y desarrollo de esas actividades, el Estado destinará recursos suficientes y creará el sistema nacional de ciencia y tecnología de acuerdo con la ley. El sector privado deberá aportar recursos para los mismos. El Estado garantizará el cumplimiento de los principios éticos y legales que deben regir las actividades de investigación científica, humanística y tecnológica. La ley determinará los modos y medios para dar cumplimiento a esta garantía.

D.2 LEY ORGÁNICA DE EDUCACIÓN

Capítulo I.

Disposiciones Fundamentales.

Artículo 2. Esta Ley se aplica a la sociedad y en particular a las personas naturales y jurídicas, instituciones y centros educativos oficiales dependientes del Ejecutivo Nacional, Estatal, Municipal y de los entes descentralizados y las instituciones educativas privadas, en lo relativo a la materia y competencia educativa.

Artículo 6. El Estado, a través de los órganos nacionales con competencia en materia Educativa, ejercerá la rectoría en el Sistema Educativo. En consecuencia:

- 1. Garantiza: El uso del idioma castellano en todas las instituciones y centros educativos, salvo en la modalidad de la educación intercultural bilingüe indígena, la cual deberá garantizar el uso oficial y paritario de los idiomas indígenas y del castellano.

Artículo 14. La educación es un derecho humano y un deber social fundamental concebida como un proceso de formación integral, gratuita, laica, inclusiva y de calidad, permanente, continua e interactiva, promueve la construcción social del conocimiento, la valoración ética y social del trabajo, y la integralidad y preeminencia de los derechos humanos, la formación de nuevos republicanos y

republicanas para la participación activa, consciente y solidaria en los procesos de transformación individual y social, consustanciada con los valores de la identidad nacional, con una visión latinoamericana, caribeña, indígena, afrodescendiente y universal. La educación regulada por esta Ley se fundamenta en la doctrina de nuestro Libertador Simón Bolívar, en la doctrina de Simón Rodríguez, en el humanismo social y está abierta a todas las corrientes del pensamiento. La didáctica está centrada en los procesos que tienen como eje la investigación, la creatividad y la innovación, lo cual permite adecuar las estrategias, los recursos y la organización del aula, a partir de la diversidad de intereses y necesidades de los y las estudiantes.

La educación ambiental, la enseñanza del idioma castellano, la historia y la geografía de Venezuela, así como los principios del ideario bolivariano son de obligatorio cumplimiento, en las instituciones y centros educativos oficiales y privados.

Capítulo II

Corresponsables de la Educación

Artículo 17. Las familias tienen el deber, el derecho y la responsabilidad en la orientación y formación en principios, valores, creencias, actitudes y hábitos en los niños, niñas, adolescentes, jóvenes, adultos y adultas, para cultivar respeto, amor, honestidad, tolerancia, reflexión, participación, independencia y aceptación. Las familias, la escuela, la sociedad y el Estado son corresponsables en el proceso de educación ciudadana y desarrollo integral de sus integrantes.

Artículo 18. Los consejos comunales, los pueblos y comunidades indígenas y demás organizaciones sociales de la comunidad, en ejercicio del Poder Popular y en su condición de corresponsables en la educación, están en la obligación de contribuir con la formación integral de los ciudadanos y las ciudadanas, la formación y fortalecimiento de sus valores éticos, la información y divulgación de la realidad histórica, geográfica, cultural, ambiental, conservacionista y socioeconómica de la localidad, la integración familia-escuela-comunidad, la promoción y defensa de la educación, cultura, deporte, recreación, trabajo, salud y demás derechos, garantías y

deberes de los venezolanos y las venezolanas, ejerciendo un rol pedagógico liberador para la formación de una nueva ciudadanía con responsabilidad social

Para desarrollar este punto se tomaron documentos oficiales legales de diferente naturaleza. De los mismos los objetivos y políticas referidos a aspectos del uso y desarrollo de tecnología educativa Venezuela.

D.3 LEY ORGÁNICA DE CIENCIA, TECNOLOGIA E INNOVACIÓN

Interés Público

Artículo 2. Las actividades científicas, tecnológicas y de innovación son de interés público y de interés general.

Artículo 3. Sujetos de esta Ley. Forman parte del Sistema Nacional de Ciencia, Tecnología e Innovación, las instituciones públicas o privadas que generen y desarrollen conocimientos científicos y tecnológicos, como procesos de innovación, y las personas que se dediquen a la planificación, administración, ejecución y aplicación de actividades que posibiliten la vinculación efectiva entre la ciencia, la tecnología y la sociedad. A tal efecto, los sujetos que forman parte del Sistema son:

- El Ministerio de Ciencia y Tecnología, sus organismos adscritos y las entidades tuteladas por éstos, o aquellas en las que tengan participación.
- Las instituciones de educación superior y de formación técnica, academias nacionales, colegios profesionales, sociedades científicas, laboratorios y centros de investigación y desarrollo tanto públicos como privados.
- Los organismos del sector privado, empresas, proveedores de servicios, insumos y bienes de capital, redes de información y asistencia que sean incorporados al Sistema.
- Las unidades de investigación y desarrollo, así como las unidades de tecnologías de información y comunicación de todos los organismos públicos.
- Las personas públicas o privadas que realicen actividades de ciencia, tecnología, innovación y sus aplicaciones.

Artículo 5. Actividades de ciencia, tecnología, innovación y sus aplicaciones. Las actividades de ciencia, tecnología, innovación y sus aplicaciones, así como, la utilización de los resultados, deben estar encaminadas a contribuir con el bienestar de la humanidad, la reducción de la pobreza, el respeto a la dignidad, a los derechos humanos y la preservación del ambiente.

Capítulo II

El Órgano Rector del Sistema

Artículo 27. De la propiedad intelectual. El Ministerio de Ciencia y Tecnología, en coordinación con los miembros del Sistema Nacional de Ciencia, Tecnología e Innovación, formulará los programas donde se establecerán las condiciones previas de la titularidad y la protección de los derechos de propiedad intelectual producto de la actividad científica, tecnológica y sus aplicaciones que se desarrollen con sus recursos o los de sus organismos adscritos.

Artículo 28. Coordinación de políticas en Propiedad Intelectual. El Ministerio de Ciencia y Tecnología, coordinará, diseñará, implementará y promoverá las políticas sobre propiedad intelectual conjuntamente con el ministerio y los demás organismos adscritos competentes en materia de protección de los derechos de propiedad intelectual de las innovaciones e invenciones producto del desarrollo de las actividades científicas, tecnológicas y sus aplicaciones concebidas en el país.

REFERENCIAS

1. Aquino A. y Zahedi A. (2007). *"El aprendizaje de las matemáticas en segundo grado de primaria por medio de dispositivos móviles"*. Trabajo de Grado. Universidad Tecnológica De La Mixteca, México. Extraído el 31 de octubre del 2014 desde: http://jupiter.utm.mx/~tesis_dig/10277.pdf
2. Beck, K. (1999) *"Extreme Programming Explained"*. Addison Wesley Professional, ISBN-13:978-0201616415, Estados Unidos.
3. Beck, K. et al. (2001). *"Principios detrás del Manifiesto Ágil"*. s.l.: Agile Alliance. Estados Unidos.
4. Billy Reynoso C. (2004). *"Métodos Ágiles en Desarrollo de Software, Introducción a la Arquitectura de Software"*. s.l.: Universidad de Buenos Aires, Argentina.
5. Borrero L. (2008). *"Enseñando a leer: Teoría, práctica e intervención"*. Grupo Editorial Norma. Bogotá, Colombia.
6. Canós J., Letelier P., Penadés M. y Sánchez E. (2003). *"Metodologías Ágiles para el Desarrollo de Software"*. s.l.: Universidad Politécnica de Valencia. Alicante, España.
7. *"Constitución de la República Bolivariana de Venezuela"*. (1999). Venezuela. Extraído el 31 de octubre del 2014 desde:
http://www.cne.gob.ve/web/normativa_electoral/constitucion/indice.php
8. cprime.com. (2013). *"Scrum Process"*. Estados Unidos. Extraído el 31 de octubre del 2014 desde: http://www.cprime.com/about/scrum_faq.html.
9. Cruz, C. (2005). *"Los genios no nacen... ¡Se hacen! Cómo programar tu mente para triunfar y ser feliz"*. Editorial Taller Éxito. 3ª Ed. Florida, Estados Unidos.

10. Cuetos V. F. (2008). *"Psicología de la lectura"*. Wolters Kluwer. España S.A. Madrid, España.
11. Díaz F. (2006). *"Principios de diseño instruccional de entornos de aprendizaje apoyados en TIC: un marco de referencia sociocultural y situado"*. Tecnología y Comunicación Educativa, 41. México. Extraído el 31 de octubre del 2014 desde: <http://investigacion.ilce.edu.mx/tyce/41/art1.pdf>
12. extremeprogramming.org. *"Extreme Programming: A gentle introduction"*. Estados Unidos. Extraído el 31 de octubre del 2014 desde: <http://www.extremeprogramming.org/>
13. Ferreyra H.A. y Pedrazzi G. (2007). *"Teorías y enfoques psicoeducativos del aprendizaje"*. Noveduc. México.
14. Galvis, A. (2001). *"Ingeniería de Software Educativo"*. Ediciones Uniandes; 3ra reimpresión de la 1ra edición. Colombia.
15. Gertner R. (2011). *"The Effects Of Multimedia Technology On Learning"*. Abilene Christian University. Texas. Estados Unidos. Trabajo de Grado de Maestría. Extraído el 31 de octubre del 2014 desde: <http://www.acu.edu/technology/mobilelearning/documents/research/effects-of-technology-on-learning.pdf>
16. git-scm.com. *"Git"*. (2012). Estados Unidos. Extraído el 31 de octubre del 2014 desde: <http://git-scm.com/>
17. google.com. *"Android Developer Tools"*. Developer.android.com Extraído el 31 de octubre del 2014 desde: <http://developer.android.com/tools/help/adt.html>
18. ingenieriadesoftware.mex.tl. *"ASD Ingeniería de Software"*. México. Extraído el 31 de octubre del 2014 desde:

http://www.ingenieriadesoftware.mex.tl/61154_ASD.html

19. *"La Enseñanza: el llamamiento más importante"*. (2000). La Iglesia de Jesucristo de los Santos de los Últimos Días. Salt Lake City, Utah, Estados Unidos.
20. ldc.usb.ve. (2006). *"Guru, Software"*. Venezuela. Extraído el 31 de octubre del 2014 desde:
<http://ldc.usb.ve/~abianc/materias/ci4713/SG-200603-Agil-RevistaMexicana.pdf>
21. *"Ley Orgánica de Ciencia, Tecnología e Innovación"*. (2005). Venezuela. Extraído el 31 de octubre del 2014 desde:
[http://www.matematica.ciens.ucv.ve/files/Normativas/Ley_Organica_de_Ciencia_Tecnologia_e_Innovacion_\(2005\).pdf](http://www.matematica.ciens.ucv.ve/files/Normativas/Ley_Organica_de_Ciencia_Tecnologia_e_Innovacion_(2005).pdf)
22. *Ley Orgánica de Educación"*. (2009). Venezuela. Extraído el 31 de octubre del 2014 desde: http://www.me.gob.ve/ley_organica.pdf
23. Madueño, L. (1997). *"Diseño de un Sistema Automatizado para la Administración de la Evaluación de los Aprendizajes a Nivel Superior"*. Trabajo de Ascenso. Universidad del Zulia, Venezuela. Extraído el 11 de enero del 2015 desde:
<http://gte2.uib.es/edutec/sites/default/files/congresos/edutec03/Ponencias/12.doc>
24. Marqués, P. (1996). *"El Software Educativo"*. Universidad Autónoma de Barcelona. España. Extraído el 31 de octubre del 2014 desde:
http://www.lmi.ub.es/te/any96/marques_software
25. Moreno, R. & Mayer, R. (2000). *"Meaningful Design for Meaningful Learning: Applying Cognitive Theory to Multimedia Explanations"*. ED-MEDIA 2000-World. Vancouver, Canada.

26. Nieto, M. (2010). *"Diseño instruccional: elementos básicos del diseño instruccional"*. Extraído el 31 de octubre del 2014 desde:
<http://es.scribd.com/doc/33372131/DISENO-INSTRUCCIONAL-TEORIAS-YMODELOS>
27. O'Connor, J. y Seymour J. (2000). *"Introducción a la Programación Neurolingüística"*. Ediciones Urano. Barcelona, España.
28. Páez B. y Sansonetti S. (2011). *"Estrategias De Enseñanza Para Estimular La Creatividad"*. Universidad Central de Venezuela. Caracas, Venezuela.
29. Papert, S., Harel, I. (1991). *"Situating constructionism"*. Constructionism, Norwood, NJ: Ablex. Estados Unidos.
30. Peláez G. y López B. (2006). *"Metodología para el Desarrollo de Software Educativo (DESED)"*. México. Extraído el 31 de octubre del 2014 desde:
<http://www.repositoriodigital.ipn.mx/bitstream/handle/123456789/5334/41-42-2.pdf?sequence=2>
31. Perkins David. (1995). *"La Escuela Inteligente: del adiestramiento de la memoria a la Educación de la Mente"*. Gedisa. Estados Unidos. Extraído el 16 de febrero del 2015 desde: <http://www.ugel05.edu.pe/ckfinder/files/la-escuela-inteligente-perkins.pdf>
32. Quero R. y Madueño L. (2006). *"Süchiki Walekerü: un ejemplo del uso de las tic en escuelas indígenas. Caso Wayuu"*. Trabajo de Grado. Universidad del Zulia, Venezuela. Extraído el 31 de octubre del 2014 desde:
http://www.scielo.org.ve/scielo.php?pid=S1316-49102006000300006&script=sci_arttext
33. Quero Y. (2000). *"Diseño de software educativo para incentivar la lectura y escritura del Wayuunaiki en los niños Wayuu"*. Proyecto Thales. Venezuela. Extraído el 16 de febrero del 2015 desde:

<http://lsm.dei.uc.pt/ribie/docfiles/txt200372921011Dise%C3%B1o%20de%20software%20educativo.pdf>

34. Sambrano, J. (2005). *"Programación Neurolingüística para todos"*. Alfadil Ediciones. 2ª. Ed. 8ª reimpresión. Caracas, Venezuela.
35. Sambrano, J. y Steiner A. (2007). *"Estrategias Educativas para docentes y padres del siglo XXI"*. Editorial Alfa. Caracas, Venezuela.
36. Sánchez M. (2002). *"Análisis de las Características y usos de los softwares educativos para niños en edad Inicial"*. Trabajo de Grado. Universidad Católica Andrés Bello. Caracas, Venezuela.
37. Scrum.org. *"Scrum"*. (2013). Estados Unidos. Extraído el 31 de octubre del 2014 desde: <http://www.scrum.org/storage/scrumguides/Scrum%20Guide%20-%20ES.pdf#view=fit>
38. una.ac.cr. *"The Agile Unified Process"*. (2014). Estados Unidos. Extraído el 31 de octubre del 2014 desde: <http://cgi.una.ac.cr/AUP/index.html>
39. Uhia A. (1969). *"Lectura y escritura"*. Editorial Voluntad. Bogotá, Colombia.
40. Villoria A., Martín G. y de las Cuevas J. (2009). *"Juegos educativos en dispositivos móviles"*. Proyecto de Sistemas Informáticos: Curso 2009/2010. Universidad Complutense de Madrid, España. Extraído el 31 de octubre del 2014 desde: http://eprints.ucm.es/11079/1/Juegos_educativos_en_dispositivos_moviles.pdf
41. www.wikipedia.org. *"Feature Driven Development"*. Estados Unidos. Extraído el 31 de octubre del 2014 desde: http://en.wikipedia.org/wiki/Feature_Driven_Development

42. [www.wikipedia.org](http://es.wikipedia.org/wiki/Herramienta_CASE). "Herramienta CASE". Wikipedia.org. Estados Unidos. Extraído el 31 de octubre del 2014 desde:
http://es.wikipedia.org/wiki/Herramienta_CASE
43. [www.wikipedia.org](http://en.wikipedia.org/wiki/Unity_(game_engine)). "Unity". Wikipedia.org. Estados Unidos. Extraído el 16 de febrero del 2015 desde: [http://en.wikipedia.org/wiki/Unity_\(game_engine\)](http://en.wikipedia.org/wiki/Unity_(game_engine))
44. Zhang E. & Loeb L. (2013). *"Mobile Applications: Games that Transform Education"*. Trabajo de Grado. Dartmouth College. New Hampshire, Estados Unidos. Extraído el 31 de octubre del 2014 desde:
<http://www.cs.dartmouth.edu/reports/TR2013-737.pdf>