Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

DESARROLLO DE UNA APLICACIÓN PARA DISPOSITIVOS MÓVILES INTELIGENTES ENFOCADA A LA PRÁCTICA DE OPERACIONES ARITMÉTICAS BÁSICAS: MATHRPG

Bryan René Alvarado Castillo
Ricardo Alfredo Illescas Alfonso
Asesorado por el Ing. Edgar Estuardo Santos Sutuj

Guatemala, mayo de 2018

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

DESARROLLO DE UNA APLICACIÓN PARA DISPOSITIVOS MÓVILES INTELIGENTES ENFOCADA A LA PRÁCTICA DE OPERACIONES ARITMÉTICAS BÁSICAS: MATHRPG

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

BRYAN RENÉ ALVARADO CASTILLO
RICARDO ALFREDO ILLESCAS ALFONSO
ASESORADO POR EL ING. EDGAR ESTUARDO SANTOS SUTUJ

AL CONFERÍRSELES EL TÍTULO DE

INGENIERO EN CIENCIAS Y SISTEMAS

GUATEMALA, MAYO DE 2018

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Pedro Antonio Aguilar Polanco
VOCAL I	Ing. Angel Roberto Sic García
VOCAL II	Ing. Pablo Christian de León Rodríguez
VOCAL III	Ing. José Milton de León Bran
VOCAL IV	Br. Oscar Humberto Galicia Nuñez
VOCAL V	Br. Carlos Enrique Gómez Donis
SECRETARIA	Inga. Lesbia Magalí Herrera López

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Pedro Antonio Aguilar Polanco
EXAMINADOR	Ing. William Samuel Guevara Orellana
EXAMINADOR	Ing. José Alfredo González Díaz
EXAMINADOR	Ing. Sergio Arnaldo Méndez Aguilar
SECRETARIA	Inga. Lesbia Magalí Herrera López

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Pedro Antonio Aguilar Polanco
VOCAL I	Ing. Angel Roberto Sic García
VOCAL II	Ing. Pablo Christian de León Rodríguez
VOCAL III	Ing. José Milton de León Bran
VOCAL IV	Br. Oscar Humberto Galicia Nuñez
VOCAL V	Br. Carlos Enrique Gómez Donis
SECRETARIA	Inga. Lesbia Magalí Herrera López

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Pedro Antonio Aguilar Polanco
EXAMINADOR	Ing. César Augusto Fernández
EXAMINADOR	Ing. Herman Igor Véliz Linares
EXAMINADOR	Ing. William Estuardo Escobar
SECRETARIA	Inga. Lesbia Magalí Herrera López

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presentamos a su consideración mi trabajo de graduación titulado:

DESARROLLO DE UNA APLICACIÓN PARA DISPOSITIVOS MÓVILES INTELIGENTES ENFOCADA A LA PRÁCTICA DE OPERACIONES ARITMÉTICAS BÁSICAS: MATHRPG

Tema que nos fuera asignado por la Dirección de la Escuela de Ingeniería en Ciencias y Sistemas, con fecha 10 junio de 2017.

Bryan René Alvarado Castillo

Ricardo Alfredo Illescas Alfonso

Guatemala, 21 de Enero de 2018

Ingeniero
Carlos Alfredo Azurdia
Coordinador de Privados y Revisor de Trabajos de Graduación
Escuela de Ciencias y Sistemas
Facultad de Ingeniería
Universidad de San Carlos de Guatemala

Ingeniero Azurdia:
Tengo el agrado de dirigirme a usted para informarle que he revisado el trabajo de graduación "Desarrollo de una aplicación para dispositivos móviles inteligentes enfocada a la práctica de operaciones aritméticas básicas: MATHRPG", realizado por los estudiantes universitarios Ricardo Alfredo Illescas Alfonso con carné 200722379 y Bryan René Alvarado Castillo con carné 200721129, quienes contaron con la asesoría del suscrito. Considero que el trabajo realizado por los estudiantes, cumple con los objetivos bajo los cuales fue planteado y cumple satisfactoriamente cada una de las actividades planificadas, por lo que procedo a aprobarlo.

Agradecimiento la atención dada a la presente,

Atentamente,

Ing. Edgar Santos
Asesor
Colegiado 5266

Edgar Santos
INGENIERO EN CIENCIAS Y SISTEMAS
Colegiado 5266

Universidad San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, 30 de enero del 2018

Ingeniero
Marlon Antonio Pérez Türk
Director de la Escuela de Ingeniería
En Ciencias y Sistemas

Respetable Ingeniero Pérez:

Por este medio hago de su conocimiento que he revisado el trabajo de graduación de los estudiantes **BRYAN RENE ALVARADO CASTILLO** con carné **200721129 y CUI 2427 53159 0101,** y **RICARDO ALFREDO ILLESCAS ALFONSO** con carné **200722379 y CUI 2332 95844 0101,** titulado: **"DESARROLLO DE UNA APLICACIÒN PARA DISPOSITIVOS MÒVILES INTELIGENTES ENFOCADA A LA PRÀCTICA DE OPERACIONES ARITMÈTICAS BÀSICAS: MATHRPG",** y a mi criterio el mismo cumple con los objetivos propuestos para su desarrollo, según el protocolo.

Al agradecer su atención a la presente, aprovecho la oportunidad para suscribirme,

Atentamente,

Ing. Carlos Alfredo Azurdia
Coordinador de Privados
y Revisión de Trabajos de Graduación

ESCUELA DE INGENIERÍA EN CIENCIAS Y SISTEMAS

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERÌA
ESCUELA DE INGENIERÍA EN
CIENCIAS Y SISTEMAS
TEL: 24188000 Ext. 1534

El Director de la Escuela de Ingeniería en Ciencias y Sistemas de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer el dictamen del asesor con el visto bueno del revisor y del Licenciado en Letras, del trabajo de graduación, ***"DESARROLLO DE UNA APLICACIÓN PARA DISPOSITIVOS MÓVILES INTELIGENTES ENFOCADA A LA PRÁCTICA DE OPERACIONES ARITMÉTICAS BÁSICAS: MATHRPG"*** *realizado por los estudiantes, BRYAN RENÉ ALVARADO CASTILLO y RICARDO ALFREDO ILLESCAS ALFONSO, aprueba el presente trabajo y solicita la autorización del mismo.*

"ID Y ENSEÑAD A TODOS"

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DIRECCION DE INGENIERIA EN CIENCIAS Y SISTEMAS

Ing. Marlon Antonio Pérez Türk
Director
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, 22 de mayo de 2018

Universidad de San Carlos
de Guatemala

Facultad de Ingeniería
Decanato

Ref.DTG.D.193.2018

El Decano de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería en Ciencias y Sistemas, al trabajo de graduación titulado: **DESARROLLO DE UNA APLICACIÓN PARA DISPOSITIVOS MÓVILES INTELIGENTES ENFOCADA A LA PRÁCTICA DE OPERACIONES ARITMÉTICAS BÁSICAS: MATHRPG,** presentado por los estudiantes universitarios: **Bryan René Alvarado Castillo y Ricardo Alfredo Illescas Alfonso,** y después de haber culminado las revisiones previas bajo la responsabilidad de las instancias correspondientes, se autoriza la impresión del mismo.

IMPRÍMASE.

Ing. Pedro Antonio Aguilar Polanco
Decano

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DECANO
FACULTAD DE INGENIERIA

Guatemala, mayo de 2018

/cc

ACTO QUE DEDICO A:

Mi madre	Mayra, por confiar en mi durante todo el transcurso de mi carrera.
Mi padre	Rene Fernando, por todas las enseñanzas que me dejó durante el tiempo que pude estar con él.
Mi pareja	Lourdes, que ha sido mi mayor soporte y motivación, no solamente durante la carrera, sino en todos los aspectos de mi vida.

Bryan René Alvarado Castillo

ACTO QUE DEDICO A:

Mis padres	Rubén y Silvia, por su amor incondicional y apoyo en todas las etapas de mi vida.
Mi esposa	Wendy, por apoyarme incondicionalmente.
Mi hijo	Lucas, por ser mi nueva fuente de inspiración y motivación para no rendirme nunca.

Ricardo Alfredo Illescas Alfonso

AGRADECIMIENTOS A:

Universidad de San Carlos de Guatemala	Por brindarme el espacio necesario para aprender y desarrollarme como profesional.
Facultad de Ingeniería	Por brindarme la oportunidad de conocer a tantas personas que contribuyeron y aportaron en mi formación profesional.
Amigo y socio	Ricardo Illescas, por todo el apoyo que me ha brindado y por ayudarme a conocer y entrar a la industria de los videojuegos.
Comunidad GameDevGT	Que ha brindado la oportunidad a tantas personas alrededor del país a conocer el desarrollo de videojuegos y mover la industria hacia delante.

Bryan René Alvarado Castillo

AGRADECIMIENTOS A:

Universidad de San Carlos de Guatemala	Por brindarme la oportunidad de cursar una carrera superior y formarme como un profesional.
Facultad de Ingeniería	Por brindarme las herramientas y los conocimientos necesarios como profesional.
Mi colega y socio	Bryan Alvarado, por el apoyo y esfuerzo durante el desarrollo de la carrera universitaria.
Mis amigos	Rodrigo y Doris Ramírez, por su apoyo en el aspecto gráfico de la aplicación desarrollada.
Comunidad y los amigos de GameDevGT	Por impulsar el desarrollo de los videojuegos en el país.

Ricardo Alfredo Illescas Alfonso

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

GLOSARIO

API	Conjunto de funciones, procedimientos y rutinas que una biblioteca expone para ser utilizado por otro software como una capa de abstracción.
Bitbucket	Servicio para alojar proyectos usando control de versiones, permite el uso de Git o Mercurial.
Bytecode	Serie de instrucciones diseñadas para ser ejecutadas de forma eficiente por un intérprete de software.
Dalvik	Máquina virtual que utiliza la plataforma para dispositivos móviles Android.
DirectX	Capa de abstracción intermedia para realizar tareas complejas relacionadas a multimedia en sistemas Microsoft Windows.
Framework	Estructura de artefactos y módulos organizados para el desarrollo de software; incluye programas, bibliotecas y un lenguaje interpretado.
Hardware	Componentes físicos y tangibles de un sistema de cómputo.

Input	Ingreso de información por parte del usuario para ejecutar instrucciones con el propósito de controlar un programa de computación.
Kernel	Constituye la parte fundamental del sistema operativo; es un componente de software que se ejecuta en modo privilegiado. Se encarga de facilitar el acceso de los programas de software al hardware de la computadora.
OpenGL	Librería de gráficos abiertos; es un estándar que define un API multilenguaje y multiplataforma para el manejo de gráficos 2D y 3D.
Rendering	En el contexto de gráficos por ordenador se refiere al proceso de generar imágenes de un modelo 2D o 3D y mostrarlos en un dispositivo de salida.
Runtime	Intervalo de tiempo en el que una aplicación se ejecuta en un ordenador.
Script	Término que se le da a un archivo de texto que contiene instrucciones para ser ejecutadas por un ordenador.
Shader	Programa que manipula los pixeles en pantalla para lograr diferentes efectos visuales.

Software	Conjunto de componentes de programas de cómputo, procedimientos, reglas, documentación y datos que forman las operaciones de un sistema de computación.
WebGL	Interface de programación de aplicaciones para renderizar gráficos 2D o 3D interactivos desde un explorador web.

RESUMEN

El aprendizaje de materias científicas, como la matemática, se percibe por los estudiantes como clases tediosas y, por ende, se crea poco interés en su aprendizaje por los métodos tradicionales.

Dada la accesibilidad en la actualidad de dispositivos móviles inteligentes, uno de los métodos para mejorar la interacción de los alumnos con la materia puede ser a través de aplicaciones interactivas, por ejemplo, juegos.

Según análisis, el aprendizaje de matemáticas por métodos tradicionales se enfoca mayormente en la reproducción de hechos; mientras que con un aprendizaje por medio de aplicaciones interactivas le permitiría al estudiante desarrollar autonomía y flexibilidad para resolver problemas de diferente índole.

El desarrollo de juegos es un área poco explorada en el ámbito tecnológico nacional; por lo cual se da una descripción de cómo llevar a cabo el desarrollo de un juego matemático en este ámbito, qué herramientas se utilizaron y cómo se modeló el juego para su utilización.

OBJETIVOS

General

Desarrollar una solución de software diseñada para dispositivos móviles con el motor de video juegos Unity 3D que permita a los usuarios practicar operaciones aritméticas básicas y proveer opciones para el nivel de dificultad y cantidad de operandos y operadores.

Específicos

1. Generar una guía que sea de utilidad a otros estudiantes para desarrollar videojuegos con el uso del motor Unity 3D en aplicaciones para dispositivos móviles que utilicen el sistema operativo Android o iOS.

2. Realizar una guía de desarrollo de una aplicación de Unity 3D, que registre el ingreso de datos desde pantallas táctiles, provea traducciones y guarde las configuraciones del usuario entre partidas.

INTRODUCCIÓN

El uso de juegos como herramientas experimentales fuera de los ámbitos tradicionales se ha desarrollado bastante durante los últimos años. La barrera que difractaba el desarrollo de juegos ha disminuido de forma considerable gracias al alto desarrollo de los motores de juegos y las herramientas similares.

Con consecuencia, el desarrollo de juegos en un país como Guatemala se ha vuelto más viable; también, está permitiendo explorar el desarrollo de juegos en otros campos que no son necesariamente juegos comerciales: juegos como arte y juegos como medio de educación.

El juego desarrollado en este trabajo es una muestra de la dirección que se puede tomar para la educación de matemáticas a nivel nacional. Cada vez los dispositivos inteligentes, como tabletas y teléfonos, se han vuelto más accesibles para la población en general, inclusive más accesible que las computadoras u otros medios electrónicos; por lo tanto, estas representan un ideal candidato para la distribución de educación interactiva.

Varios estudios, también, han demostrado que el contacto constante con juegos les permite a los niños desarrollar habilidades que usualmente no se fomentan en métodos tradicionales de educación, por ejemplo, pensamiento estratégico, resolución de problemas; coordinación visomotriz y mejora de la retención de conceptos.

En los siguientes capítulos se describen los conceptos y las teorías usadas de base para crear MathRPG; también, las técnicas y herramientas utilizadas.

1. ESTUDIO, TECNOLOGÍA E IMPACTO EN GUATEMALA

1.1. Impacto en Guatemala

"A pesar de que la enseñanza de la Matemática ha cobrado importancia en los últimos años, los resultados obtenidos por los estudiantes demuestran que aún hay retos por superar. En las evaluaciones nacionales realizadas en el 2010 el 46,26 % de estudiantes de primero y el 48,67 % de tercero primaria de centros educativos públicos alcanzan el logro esperado. Esto significa que más de la mitad no alcanzan el nivel de logro, quedándose a nivel de reproducción, de definiciones y cálculos sin lograr llegar al pensamiento matemático y hacer generalizaciones para resolver problemas de la vida cotidiana."[1]

Tomando en cuenta que cada vez es mayor la accesibilidad de los dispositivos móviles que tienen los estudiantes, el juego podría incrementar considerablemente la exposición a las matemáticas; además, por la forma en que será presentada, no solo mejorará su aprendizaje, también, la posibilidad de hacerlo una actividad recreativa para el estudiante. Representará una actividad complementaria a la enseñanza impartida en los estudios convencionales.

[1] *Programa contemos juntos*. https://www.mineduc.gob.gt/CONTEMOS_JUNTOS/ documents/Plan_ Programa_Contemos_Juntos.PDF. Consulta: 6 de abril de 2017.

1.2. iOS

1.2.1. Historia

El sistema operativo fue mostrado por primera vez al público en la Macworld Conference and Expoel 9 de enero de 2007 y fue liberado en junio de ese mismo año. Cuando fue mostrado al público, Steve Jobs decía que el iPhone corría OS X, su sistema operativo de escritorio; pero una vez fue lanzado, el sistema operativo fue renombrado como iPhone OS. Inicialmente, no tenía soporte para aplicaciones de terceros, pensando que los desarrolladores podrían crear aplicaciones con aplicaciones web que serían utilizadas en su navegador web Safari. Sin embargo, esto no funcionó y en marzo de 2008 fue anunciado el primer kit de desarrollo del iPhone.

En julio de 2008 se lanzó la iOS App Store, con alrededor de 500 aplicaciones disponibles. En tan solo 2 meses el número de aplicaciones creció a 3 000, en 6 meses a 15 000 y para junio de 2009 había disponibles alrededor de 50 000 aplicaciones. En enero de 2017 ya se encontraban disponibles alrededor de 2,2 millones de aplicaciones en la tienda.

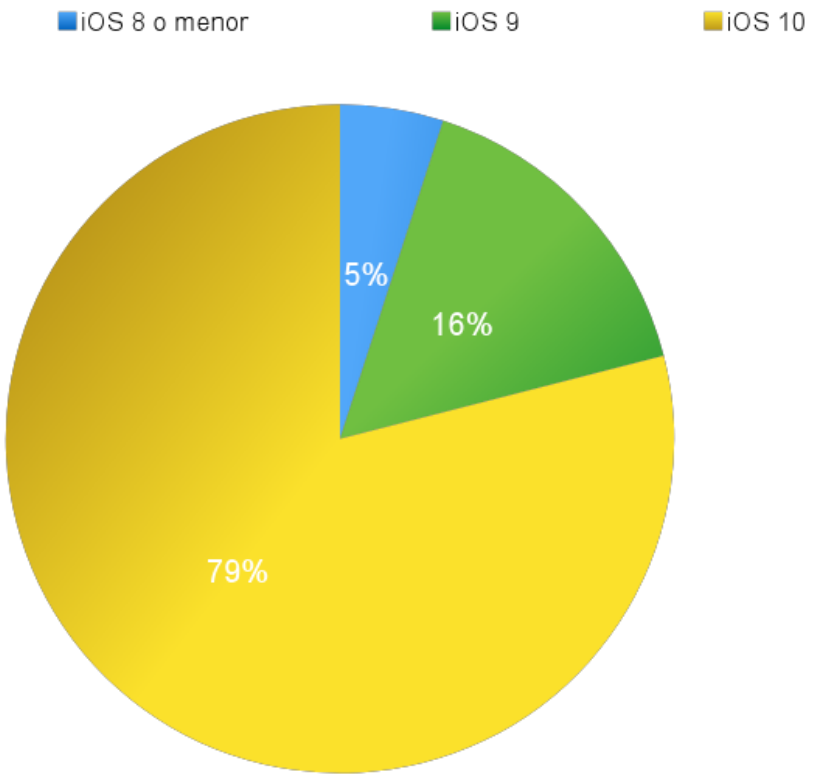
En junio de 2010, Apple cambió el nombre de iPhone OS a simplemente iOS.

En septiembre de 2007, Apple anunció el lanzamiento del iPod Touch, que tenía la mayoría de capacidades del iPhone menos las características de telefonía. En enero de 2010, se anunció el iPad, que contaba con una pantalla de mayor tamaño que las del iPhone y iPod Touch y estaba diseñada para lectura, navegación de la web y consumo de videos.

Durante la existencia del sistema operativo se han lanzado 10 versiones mayores del mismo.

El porcentaje de uso de cada versión del sistema operativo para febrero de 2017 se encontraba de la siguiente manera:

Figura 1. **Porcentaje de uso en versiones de iOS**

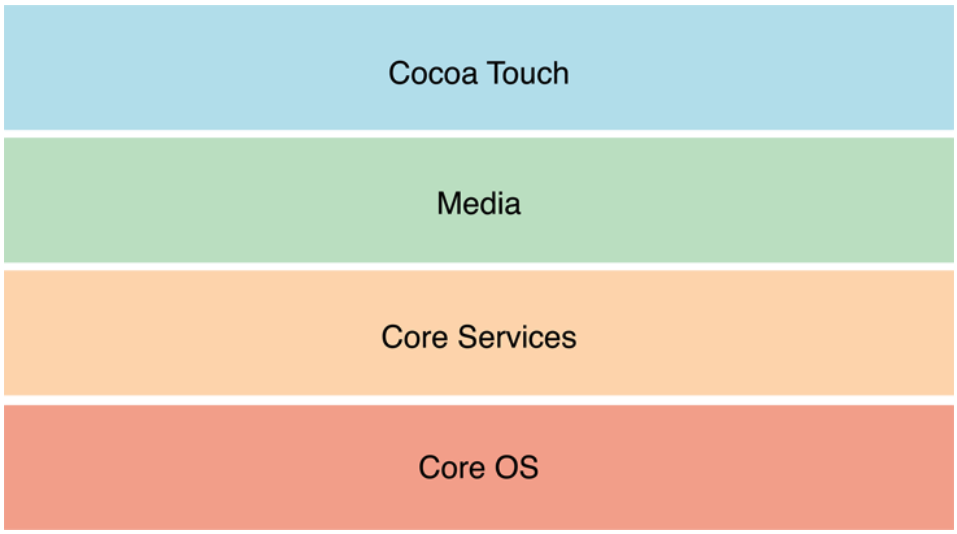


Fuente: elaboración propia.

1.2.2. Arquitectura

En su nivel más alto, iOS funciona como un intermediario entre el hardware y las aplicaciones las cuales se comunican con el hardware a través de un conjunto de interfaces de sistema.

Figura 2. **Arquitectura iOS**



Fuente: elaboración propia.

1.2.2.1. Cocoa Touch

Esta capa contiene los *frameworks* para construir aplicaciones de iOS. Estos definen la apariencia de la aplicación. También, proveen la infraestructura básica de la aplicación y soporta tecnologías clave como las entradas táctiles, notificaciones y varios servicios del sistema de alto nivel.

1.2.2.2. Media Layer

Esta capa contiene las tecnologías de gráficos, audio y video que se utilizan para implementar cualquier tipo de multimedia en las aplicaciones. Esta capa ayuda a facilitar la construcción de aplicaciones que se vean y suenen bien.

1.2.2.3. Core Services

Esta capa contiene los servicios del sistema fundamentales para aplicaciones que definen todos los tipos básicos que usan todas las aplicaciones. También, contiene tecnologías individuales para soportar funcionalidad como locaciones, medios sociales y redes.

1.2.2.4. Core OS

Esta capa contiene las funcionalidades de bajo nivel sobre las que están construidas las demás tecnologías de las otras capas. Aun cuando estas no estén siendo utilizadas directamente por una aplicación, lo más seguro es que estén siendo utilizadas por alguna de las tecnologías implementadas en la aplicación. Entre algunas de sus funcionalidades están el acelerómetro, accesorios externos, extensiones de red, seguridad, acceso al sistema de archivos, etc.

1.3. Android

Android es un sistema operativo para dispositivos móviles. En sus inicios fue desarrollado por Android, Inc.

Está basado en el *kernel* Linux y contempla como principal método de entrada una interfaz táctil con soporte para gestos para manipular objetos en pantalla, junto con un teclado virtual para el ingreso de datos.

1.3.1. Historia

En 2003, Andy Rubin, Rich Miner, Chris White y Nick Sears fundaron Android Inc., con el objetivo de desarrollar un sistema operativo para dispositivos móviles basado en Linux.

En 2005 fue adquirido por Google y en 2007 se crea la Open Handset Alliance, un grupo conformado por desarrolladores de hardware, software y operadores de servicios telefónicos con el fin de crear estándares abiertos para dispositivos móviles. Durante el anuncio de la formación de la Open Handset Alliance, se presenta Android por primera vez.

La primera versión del sistema operativo Android, nombrado Apple Pie, es lanzada en 2007 y los primeros dispositivos móviles Android llegan al mercado en el año 2008.

1.3.2. Arquitectura

Android es una pila de software de código abierto basado en Linux diseñado para una amplia cantidad de dispositivos de todo tipo de forma y tamaño.

1.3.2.1. El *kernel* Linux

La base de la plataforma Android es el *kernel* Linux. Esto es primordial para manejar funcionalidades de bajo nivel, como el manejo de memoria.

El uso del *kernel* Linux le permite a Android tomar ventaja de las características de seguridad y permite a los fabricantes desarrollar controladores para el *kernel* tan conocido.

1.3.2.2. Capa de abstracción de hardware

Capa de abstracción de hardware que provee un estándar de interfaces que exponen características de hardware al API de alto nivel de Java. La capa

de abstracción de hardware consiste en múltiples módulos de librerías, los cuales implementan una interfaz para un componente de hardware específico, como acelerómetros y giroscopios.

Cuando un API desea utilizar un dispositivo, el sistema carga un módulo de librería que permite utilizar el componente de hardware.

1.3.2.3. Android RunTime

Para dispositivos con la versión 5.0 de Android (nivel de API 21) o avanzado, cada aplicación opera como un proceso independiente y tiene su propia instancia del tiempo de ejecución de Android. ART soporta la ejecución de máquinas virtuales múltiples en dispositivos de poca memoria ejecutando archivos DEX, un formato de *bytecode* diseñado específicamente por Android y que está optimizado para ser muy eficiente en el uso de memoria.

Algunas de las características principales de ART incluyen la compilación antes de tiempo (AOT) y justo a tiempo (JIT), recolección de basura optimizada (GC), soporte para depuración, un perfilador de muestras, diagnóstico de excepciones detalladas y reporte de fallos.

“Muchos componentes y servicios de Android, como ART y HAL, se crean a partir de código nativo que requiere bibliotecas nativas escritas en C y C++. La plataforma Android proporciona un API de Java para exponer la funcionalidad de algunas de estas bibliotecas nativas a las aplicaciones.” [2]

[2] Android. https://developer.android.com/guide/platform/index.html. Consulta: 6 de abril de 2017.

Por ejemplo, es posible acceder a OpenGL ES a través de la API Java OpenGL de Android para agregar soporte para dibujar y manipular gráficos 2D y 3D en las aplicaciones.

1.3.2.4. Estructura de API de Java

El completo conjunto de características de Android está disponible a través de APIs escritas en el lenguaje Java. Estas APIS representan el fundamento para crear aplicaciones Android al simplificar el reuso de componentes y servicios.

Estos componentes incluyen:

- El sistema de vistas con el cual es posible construir interfaces de usuario, incluyendo listas, retículas, cuadros de texto, botones y hasta un explorador web embebido.

- El manejador de recursos provee acceso a recursos como localización de secuencias de caracteres, gráficos y archivos de diseño.

- El manejador de actividades, que controla el ciclo de vida de la aplicación.

Android incluye un conjunto de aplicaciones para correo electrónico, mensajes SMS, calendarios, navegación por internet, contactos y más.

1.4. Motor de videojuegos

Un motor de videojuegos es un *framework* diseñado para la creación de videojuegos. La funcionalidad principal del motor de videojuegos es proporcionar a los desarrolladores herramientas de renderizado para gráficos 2D o 3D, simulación de física, detección y resolución de colisiones, audio, *scripting*, animación, redes. El proceso de desarrollo de juegos se economiza en gran manera al usar y adaptar el mismo motor de juegos para diferentes proyectos o para desarrollar juegos multiplataforma.

Antes de la creación de los motores de videojuegos estos eran escritos para una plataforma específica y raramente el código era reutilizado para diferentes proyectos; conforme la complejidad de los juegos fue incrementando fue necesario separar los componentes.

El término motor de juegos cobró auge en la década de 1990, con juegos como Doom. Doom contaba con una arquitectura bastante desacoplada, que hacía una separación entre sus componentes principales de software (sistema de renderizado de gráficos, detección de colisiones o el sistema de audio) y los elementos de arte, niveles del juego y las reglas del juego. El valor de separar todos los componentes fue evidente cuando los desarrolladores fueron capaces de crear nuevos juegos solo remplazando arte, niveles, reglas y haciendo modificaciones leves a los componentes principales.

Un motor de videojuegos está formado de varios componentes.

1.4.1. Programa principal del juego

La lógica del juego debe ser implementada por ciertos algoritmos, es distinta de cualquier trabajo de renderizado, sonido o entrada de información.

1.4.2. Motor de renderizado

Se define renderizado como el proceso en el que una descripción geométrica de un objeto es convertida a una representación en un plano bi dimensional que lo hace parecer real.

El motor de renderizado genera gráficos animados según el método soportado por el sistema: rasterización, trazado de rayos u otra técnica.

En lugar de programarse y compilarse para ejecutarse en la CPU o GPU directamente, la mayoría de los motores de renderización se construyen sobre una o múltiples interfaces de programación de aplicaciones, como Direct3D u OpenGL que proporcionan una abstracción de software de la unidad de procesamiento de gráficos (GPU).

Las bibliotecas de bajo nivel como DirectX, Simple DirectMedia Layer (SDL) y OpenGL también se utilizan comúnmente en los juegos, ya que proporcionan acceso independiente del hardware a otro hardware, como dispositivos de entrada, tarjetas de red y tarjetas de sonido.

1.4.3. Audio Engine

El motor de audio es un componente que consiste en algoritmos relacionados a la generación y reproducción de sonido. Puede realizar cálculos

en el CPU o en un circuito integrado específico para la aplicación (ASIC). También, se dispone de abstracciones de APIS como OpenAL, SDL Audio, XAudio2, WebAudio y similares.

1.4.4. Motor de físicas

El motor de físicas es responsable de generar una representación acertada de las leyes de la física dentro de la aplicación.

El control de colisiones y de aplicaciones de fuerzas permite desplazar objetos a través del espacio y provee al desarrollador crear funciones que interactúan con el sistema de físicas.

Además, el motor permite la suficiente personalización para ser capaz de simular diferentes tipos de modelos físicos.

1.4.5. Inteligencia artificial

El componente de AI es un módulo especial el cual es diseñado y escrito por ingenieros de software especialistas en el desarrollo de rutinas de AI.

1.5. Unity Game Engine

Unity es un motor de videojuegos desarrollado por la compañía Unity Technologies, el cual es principalmente usado para desarrollo de videojuegos y simulaciones para computadoras, consolas de sobremesa y dispositivos móviles.

Unity es motor multipropósito y por ende soporta tanto gráficos 2D como 3D, funcionalidad *drag and drop* y *scripting* a través de tres lenguajes de programación.

1.5.1. Historia

Los inicios de Unity pueden rastrearse hasta una publicación en un foro de internet de OpenGL para Mac OS en mayo de 2002, cuando Nicholas Francis solicitaba ayuda con un sistema de *shaders* que trataba de implementar en un motor de juegos de su autoría. A esta publicación responde Joachim Ante y comienzan a colaborar en un sistema de *shaders* que sea compatible con los motores de juegos de ambos programadores. Con el tiempo decidieron abandonar sus desarrollos independientes y enfocarse en crear un solo motor de juegos de forma colaborativa. David Helgason se unió tiempo después como el tercer programador del proyecto.

Inicialmente, deseaban desarrollar un juego y licenciar la tecnología, un juego que probara lo que el motor era capaz de hacer; al final decidieron no hacer juegos, en cambio, le enfocaron solamente en desarrollar una herramienta para realizar juegos.

La primera versión pública de Unity solo permitía desarrollar juegos para Mac OS X, la versión 1.1 agregó soporte para exportar a plataformas basadas en Windows.

Actualmente, Unity permite exportar juegos a PC, Mac, Linux, consola de videojuegos, WebGL.

Con el tiempo muchos desarrolladores comenzaron a adoptar el motor Unity para realizar sus juegos, pero esto requería que usaran ordenadores Macintosh para ejecutar el editor; el equipo de Unity comenzó a trabajar en agregar soporte para plataformas Windows; aunque esto significó tener que reescribir el editor para hacerlo independiente de plataforma; el nuevo editor compatible con Windows estuvo disponible en la versión 2.5 de Unity, lanzada en 2009 en la Game Developers Conference.

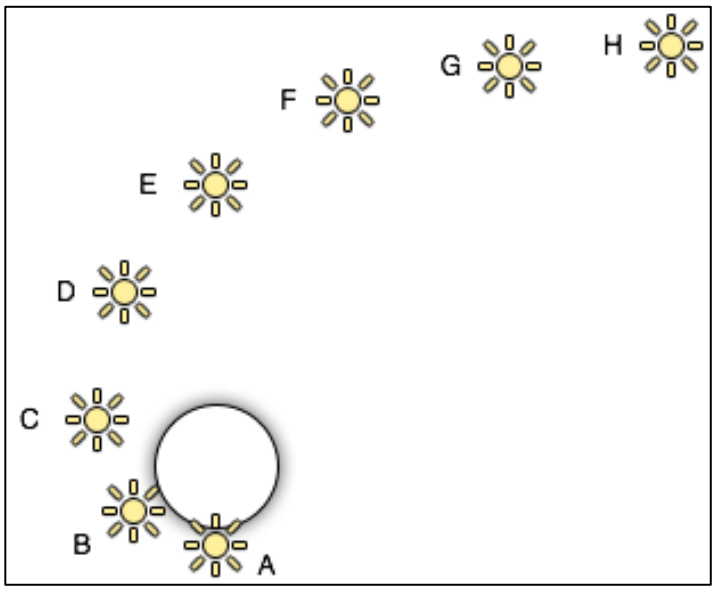
1.5.2. Arquitectura Unity Game Engine

1.5.2.1. Pipeline de renderizado

Una de las principales características de Unity es que permite elegir entre dos tipos de renderizado: renderizado hacia delante (*forward rendering*) y sombreado diferido (*deferred shading*); la diferencia entre este tipo de renderizado es la generación de luces, mostrar luces y sombras en tiempo real; es una tarea bastante intensiva, que requiere muchos cálculos y ciclos de procesamiento.

El renderizado hacia delante renderiza un objeto en pantalla en una o más pasadas, dependiendo en la cantidad de luces que afectan al objeto; hasta cuatro luces pueden ser calculadas por vértice; el resto son calculadas usan Spherical Harmonics, que es más rápido, pero solo es una aproximación.

Figura 3. **Iluminación sobre un objeto tridimensional**

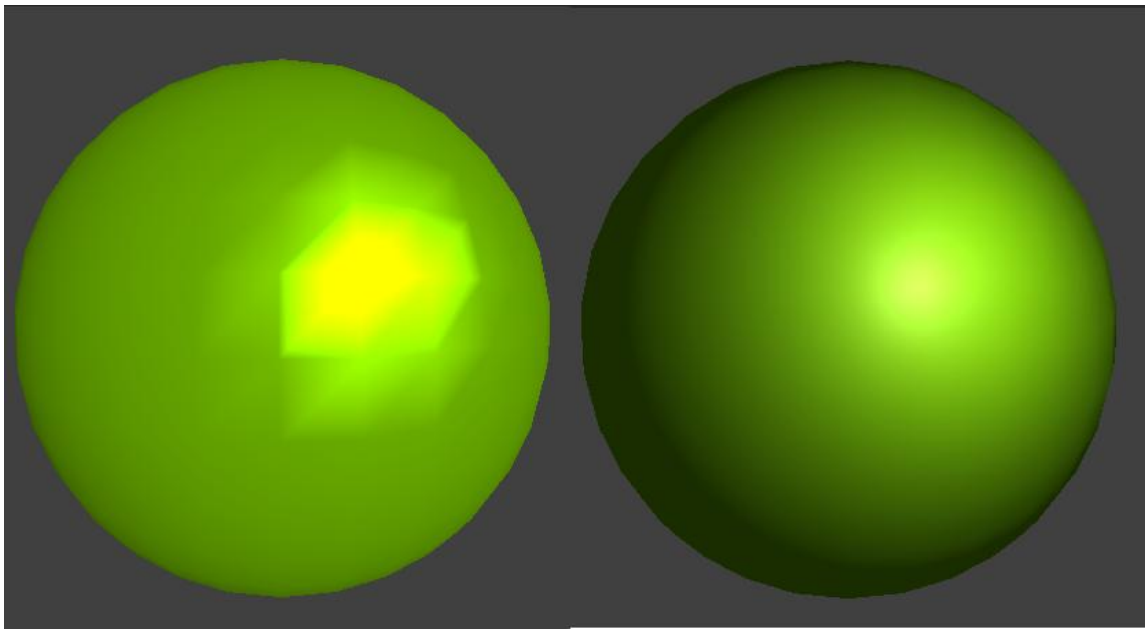


Fuente: *Unity 3D*. https://unity3d.com/es. Consulta: 29 de diciembre de 2017.

El sombreado diferido no impone un límite en la cantidad de luces que pueden afectar a un objeto. Todas las luces son evaluadas por pixel, de forma que interactúan sobre mapas normales de forma correcta y todas las luces pueden tener cookies y sombras. Tiene la ventaja de que la carga de procesamiento es proporcional al número de píxeles que se iluminan, determinado por el tamaño del volumen de luz en la escena, independientemente, a cuantos objetos se ilumine; de esta forma es posible mejorar el rendimiento usando luces pequeñas.

Un comportamiento negativo de este método es que no hay soporte para *anti-aliasing* y que no se pueden manejar transparencias, los objetos transparentes son renderizados con *forward rendering*.

Figura 4. **Diferencia entre sombreado diferido y renderizado hacia delante**



. Fuente: elaboración propia, empleando Unity.

1.5.2.2. Ciclo de ejecución de una aplicación de Unity

En Unity hay un número de métodos que son ejecutados en un orden determinado mientras la aplicación se encuentra en funcionamiento; la mayoría de estos métodos caen en 3 categorías principales: al momento de inicializar un objeto, mientras el objeto está activo y al de comisionar un objeto.

1.5.2.2.1. Al inicializar

Permite definir el código que solo será ejecutado cuando un objeto es creado en la escena; esto es principalmente utilizado para inicializar variables,

definir valores predeterminados, posicionar objetos en posiciones iniciales, cargar dependencias y buscar otros objetos en la escena.

Para los objetos colocados en una escena, los métodos de inicialización OnEnable, Awake, Start son llamados antes de que cualquier llamada durante la ejecución sea efectuada.

1.5.2.2.2. Durante la ejecución

Permite llevar un récord de la lógica de juego, las interacciones entre objetos, animaciones, movimientos de cámara y demás actividades que deben ejecutarse de manera constante durante el juego. Unity maneja una serie de métodos que se llaman mientras los objetos están activos en escena; el orden de estos métodos permite efectuar diferentes lógicas según sea conveniente.

FixedUpdate es llamado de forma frecuente. Todas las iteraciones del motor de físicas ocurren después de una llamada al FixedUpdate.

Update se llama una vez por cuadro de ejecución; este método está atado a la velocidad con la que se procesa el juego por lo que el tiempo de llamadas entre ejecución puede variar.

LateUpdate es ejecutado una vez por cuadro de ejecución cuando el método Update ha terminado. Uno de los usos comunes de este método es para manejar las actualizaciones de la cámara.

1.5.2.2.3. Al finalizar ejecución

Cuando un objeto es eliminado de la escena, el método OnDestroy es ejecutado. Cuando la aplicación es finalizada, el método OnApplicationQuit es llamado en todos los objetos activos en la escena.

1.5.2.3. Control de entrada de datos

La clase input permite tener acceso a todos los dispositivos configurados y conectados al sistema donde la aplicación se está ejecutando.

Unity tiene soporte para los dispositivos de entrada más comunes utilizados en videojuegos, teclados, ratón, mandos de juego. Es posible, también, recibir datos desde pantallas táctiles y de sensores sensibles al movimiento, como acelerómetros y giroscopios los cuales son bastante comunes en dispositivos móviles.

En dispositivos móviles, como iOS y Android es posible reconocer múltiples puntos interactuando con la pantalla de forma simultánea; estos puntos son denominados ‘toques’ y permiten obtener el estado de cada dedo que tocó la pantalla durante el último cuadro de actualización.

Los puntos detectados por la aplicación son almacenados en un arreglo de datos; estos puntos son una coordenada en un espacio bidimensional basado en la posición de la pantalla; de esta forma es posible detectar gestos realizados por el usuario y reaccionar dentro de la aplicación de forma adecuada.

Dispositivos Apple, como el iPhone, son capaces de reconocer hasta un máximo de cinco puntos en pantalla de forma simultánea. Los dispositivos Android no tienen un límite unificado de puntos de detección simultáneos lo cual los hace variar entre poder reconocer desde un máximo de dos puntos en los dispositivos más antiguos hasta cinco, en los dispositivos más modernos.

De forma que el dispositivo se mueve, el hardware del acelerómetro reporta la aceleración lineal en relación a los tres ejes principales en un espacio tridimensional; esta información es accesible por la aplicación y permite detectar cambios en la orientación del dispositivo.

1.6. Herramientas

1.6.1. Bitbucket

Bitbucket es un servicio de alojamiento en la red, propiedad de la empresa Atlassian, usado para código fuente y desarrollo de proyectos que utilizando los sistemas de control de versiones Mercurial o Git. Este ofrece tanto planes comerciales y cuentas gratuitas. A las cuentas gratuitas se les ofrece una cantidad ilimitada de repositorios privados, con un límite en la cantidad de usuarios que puede tener cada repositorio.

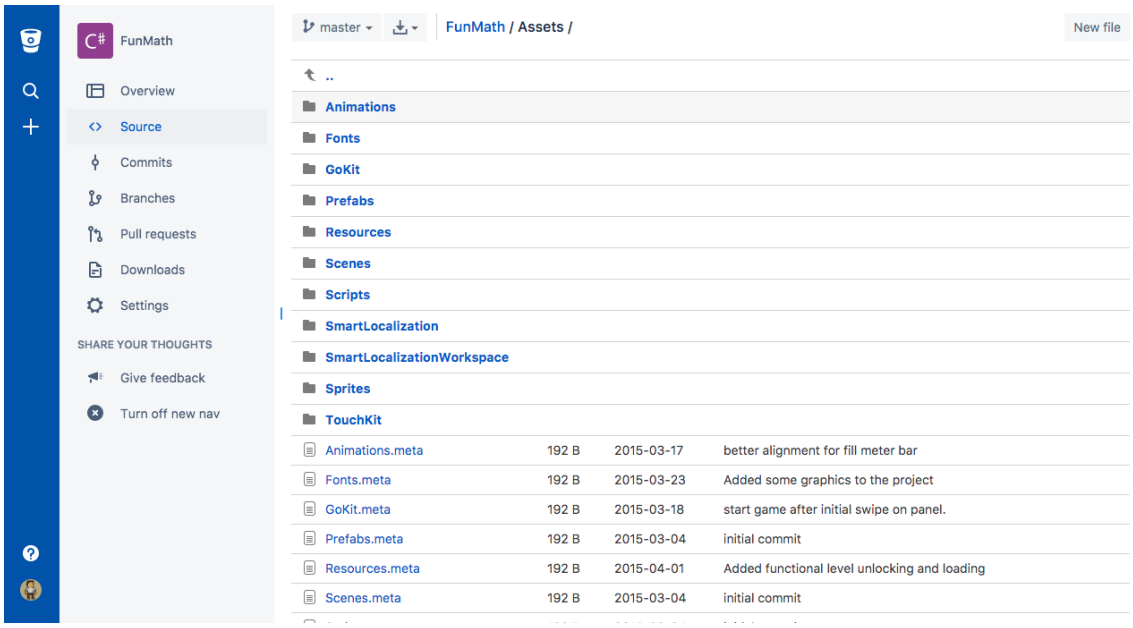
Está desarrollado en el lenguaje de programación Python con el marco de desarrollo web Django.

Algunas de las características de Bitbucket son:

- Revisión de código y comentarios
- Verificación de 2 pasos

- Búsqueda de código
- Documentación
- Paginas estáticas
- Integraciones y otros agregados

Figura 5. **Interfaz web de Bitbucket**



Fuente: elaboración propia, empleando Unity.

1.6.2. Microsoft Visual Studio

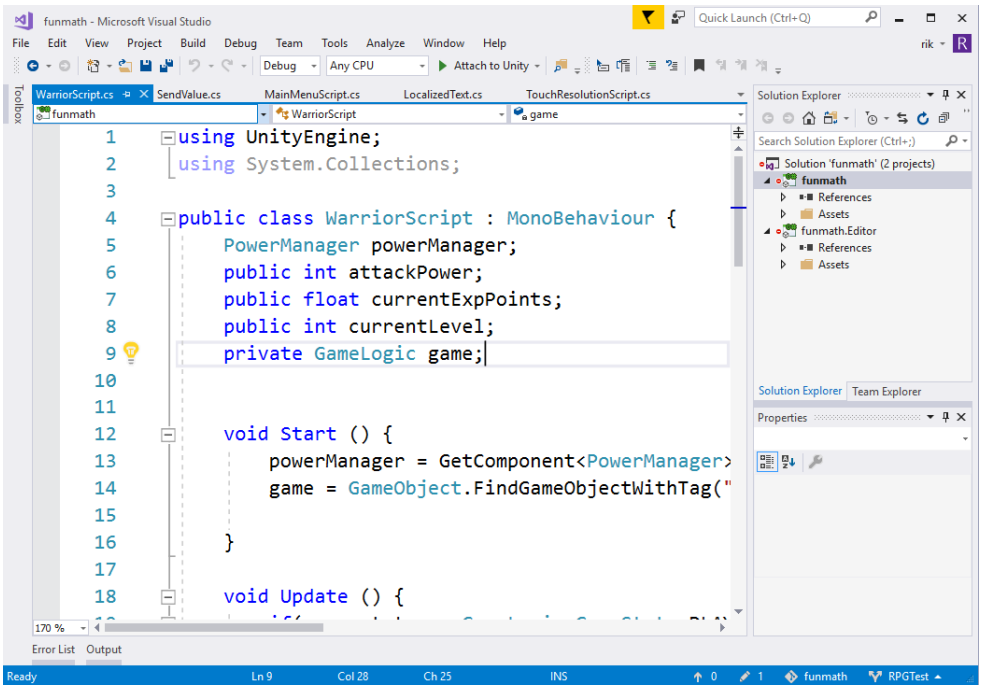
Microsoft Visual Studio es un ambiente integrado de desarrollo. Es utilizado para desarrollar tanto programas para computadoras Windows como para el desarrollo de sitios web, aplicaciones web, servicios web y aplicaciones móviles.

Visual Studio incluye un editor de código con un componente de compleción de código, así como refactorización de código. El depurador integrado funciona tanto a nivel de fuente como a nivel de máquina. Otras herramientas que vienen incluidas son un perfilador de código, un constructor de interfaces de usuario, un diseñador de clases y un diseñador de esquemas de bases de datos. También posee un sistema para adicionar funcionamiento en casi cualquier nivel, el cual incluye soporte para el control de versiones en varios formatos.

Soporta alrededor de 36 diferentes lenguajes de programación y permite al editor de código y al depurador soportar casi cualquier lenguaje de programación, si se provee un servicio específico del lenguaje. Algunos de los lenguajes que incluye son C, C++, VB.NET, C#, F# y TypeScript. Soporte para otros lenguajes como Python, Ruby y Node.js están disponibles a través de servicios de lenguaje instalados por separado.

Microsoft provee una versión gratis de Visual Studio llamada la edición de comunidad que soporta el sistema de adiciones y está disponible sin costo alguno.

Figura 6. **Microsoft Visual Studio**



Fuente: elaboración propia, empleando Unity.

1.6.3. Git

Git es un sistema de control de versiones gratuito y de código abierto, diseñado para manejar proyectos de todo tamaño con velocidad y eficiencia.

1.6.3.1. Ramas y combinación de ramas

La característica de Git que lo hace resaltar de otros sistemas de control de versiones es su modelo de ramas.

Git permite y alienta el tener múltiples ramas locales que pueden ser completamente independientes entre estas. La creación, unión y supresión de esas líneas de desarrollo toma segundos.

Esto permite tener beneficios como:

- Cambio de contextos sin fricción
- Líneas de código con roles específicos
- Un flujo de trabajo basado en características
- Experimentación desechable

1.6.3.2. Distribuido

Otra de las características importantes de Git, como otros sistemas de control de versiones, es que es distribuido, lo que significa que en vez de revisar solo la última versión del código fuente, se hace un clon completo del repositorio completo.

Esto permite que, aun utilizando un flujo de trabajo centralizado, cada usuario pueda tener un respaldo completo del servidor principal. Cada una de estas copias puede ser enviada al servidor principal para reemplazar el código fuente en caso de corrupción o algún otro inconveniente con ésta.

1.6.3.3. Garantía de datos

El modelo de datos de Git asegura la integridad criptográfica de los datos en el proyecto. Cada archivo y cada commit tiene una suma de comprobación y es manejado por esta misma suma cuando se revisa de regreso. Es imposible obtener algo de Git que no sea la misma data que se puso dentro del repositorio.

2. TEORÍAS QUE RESPALDAN EL DESARROLLO

2.1. Teoría de la carga cognitiva

2.1.1. Principales factores dependientes

Rendimiento, que puede estar sujeto a adquisición de conocimiento, aprendizaje, adquisición de esquemas y resolución de problemas.

2.1.2. Principales factores independientes

Esfuerzo mental, carga mental, subcompuesta por carga intrínseca, carga desconectada y carga conectada.

2.2. Descripción de la teoría

La teoría de la carga cognitiva fue en su mayoría desarrollada por John Sweller en 1988 como parte de su investigación sobre resolución de problemas. La teoría propone que el aprendizaje puede ser mejorado por la presentación de la información. La teoría asume una limitada memoria en trabajo y una memoria de largo plazo virtualmente ilimitada.

Esquemas, que categorizan información por la manera en la que será utilizada, son adquiridos con el tiempo y repetida exposición a problemas relacionados, son automatizadas como reglas, y son guardadas en la memoria a largo plazo para ser utilizadas cuando son necesarios. Aunque la memoria funcional ha demostrado procesar solo un número limitado de objetos, trata los

esquemas (que pueden ser bastante detallados y complicados y representan una gran cantidad de información) como un solo objeto. Por lo tanto, estructurar información para que el estudiante pueda desarrollar esquemas y automatizar reglas para guardar en su memoria a largo plazo mejora la adquisición y rendimiento de aprendizaje.

Hay varios efectos y técnicas que contribuyen a la carga cognitiva y la habilidad del sujeto de adquirir esquemas y automatizaciones. Los primeros trabajos de Sweller se enfocan en el mejor método para presentar problemas y examinaba medios y fines de análisis, problemas sin metas, ejemplos trabajados y terminación de problemas. Uno de los efectos más usados por Sweller era el efecto de modalidad, el cual plantea que la información presentada en forma auditiva y visual incrementa la memoria funcional.

La facilidad con la que la información es procesada por la memoria funcional es la principal preocupación de la teoría. La carga de memoria funcional puede ser afectada por la naturaleza intrínseca del material, o alternativamente, por la manera en que el material es presentado o las actividades requeridas por los estudiantes.

Una vez un esquema es construido, los elementos interactivos son incorporados al esquema y no necesitan ser considerados individualmente dentro de la memoria funcional. El esquema puede actuar como un solo elemento dentro de la memoria funcional e impondrá demandas mínimas sobre la misma, especialmente, si es automatizada una vez construido el esquema.

2.3. Relación de la teoría con el proyecto

La accesibilidad de dispositivos digitales ha aumentado considerablemente en la última década, lo que ha generado una interacción estrecha entre las personas, incluyendo niños y estos dispositivos. Los niños han incrementado su uso de dispositivos móviles, particularmente, los juegos. Por tal razón, al utilizar un juego como herramienta educativa, se puede mejorar el rendimiento de aprendizaje que poseen los niños, con actividades interactivas que crearán esquemas mentales mucho mayores a los medios convencionales de aprendizaje, sobre todo cuando se trata de matemáticas; se le da la oportunidad al niño de absorber este tipo de conocimiento de una forma más automatizada y, al mismo tiempo, interactiva; recibe retroalimentación de forma audiovisual.

Según el estudio de uso de medios de la organización Common Sense Media en 2013 se estima que el 75 % de los niños en Estados Unidos ya han hecho uso de un dispositivo móvil a la edad de 8 años.

2.4. Delineamiento del problema

En evaluaciones nacionales realizadas en 2010, el 46 % de estudiantes de primero y el 48 % de tercero primaria de centros educativos públicos alcanzan el logro esperado. Esto significa que más de la mitad de alumnos no alcanzan el nivel deseado, quedándose a nivel de reproducción de definiciones y cálculos sin lograr llegar al pensamiento matemático y hacer generalizaciones para resolver problemas de la vida cotidiana.

El análisis de estos resultados permite concluir que en las escuelas, el aprendizaje de las Matemáticas se enfoca en la reproducción de hechos más

que en desarrollar autonomía y flexibilidad para resolver problemas de diferente índole. Esto limita a los estudiantes la capacidad de intervenir, proponer y responder a situaciones que se le presentan en el contexto en que se desempeña de forma creativa y competente.

En 2013, como parte del programa Contemos Juntos del Ministerio de Educación, se les consultó a estudiantes acerca del tiempo que intervienen en su casa para realizar tareas de matemática, entre el 50 % y 60 % de los estudiantes de primero primaria respondió media hora, mientras que el 56,4 % de los de tercero primaria indicó que menos de una hora.

La falta de práctica en esta área es algo que no solo incrementa su dificultad, también, limita la creatividad y la habilidad para la resolución de problemas en la vida diaria.

2.4.1. Mercado objetivo

La aplicación está dirigida a niños entre 7 y 11 años, quienes todavía están en el proceso de aprendizaje de las operaciones algebraicas básicas, que son con las que contará la aplicación en su primera versión. Sin embargo, cualquier persona, indiferentemente de la edad, podrá hacer uso de la aplicación, ya sea de manera recreativa o como herramienta de práctica de dichas operaciones.

2.4.2. Solución propuesta

El desarrollo de un juego para dispositivos móviles que tenga el propósito de aumentar la accesibilidad y usabilidad de la práctica de operaciones básicas algebraicas. En este juego se presentarán varias opciones de práctica, también,

una interfaz intuitiva para los niños; será un ambiente controlado donde podrían practicar sus operaciones para mejorar la absorción del aprendizaje matemático que estén llevando en ese momento.

2.5. *Benchmarking* de la aplicación

Esta sección presenta un conjunto de aplicaciones que tienen relación con el tema de práctica de aritmética en un dispositivo móvil; cada una tiene ciertos puntos que representan ventajas y desventajas; en algunos casos la funcionalidad se asemeja a nuestra solución, pero no satisface en su totalidad la problemática planteada.

2.5.1. Brain Math Game

Es una aplicación para dispositivos móviles que permite realizar el desafío de Einstein que prueba que tan bien el cerebro procesa números. El juego muestra una serie de números en pantalla; luego, muestra cuatro posibles respuestas; el usuario debe seleccionar cuál de las respuestas corresponde a la suma de los números mostrados previamente.

2.5.1.1. Funciones principales

- Einstein’s Math Game

Einstein’s Math Game reta al jugador a realizar sumas con distintas cantidades de números enteros desde su dispositivo móvil Android; si el jugador responde correctamente, el juego muestra una nueva operación a realizar.

Al iniciar el juego se puede definir la dificultad (fácil, medio, difícil); estas opciones hacen variar la cantidad de números que aparecerán en pantalla.

2.5.1.2. Ventajas

- El modo del juego es fácil de comprender
- Puede variarse la dificultad de las sumas

2.5.1.3. Desventajas

- Solo pueden realizarse sumas
- Interfaz de usuario poco atractiva

2.5.2. Math Workout

Es una aplicación para dispositivos móviles Android que permite a los usuarios resolver series de ejercicios aritméticos.

2.5.2.1. Funciones principales

- Addition & Subtraction / Multiplication & Division

Este modo permite jugar hasta completar 20 ejercicios divididos en 2 grupos, suma y resta, multiplicación y división. Al finalizar los ejercicios, se muestra al usuario información de tiempo en resolver el reto, intentos fallidos y se le da calificación basada en su desempeño.

- The Brain Cruncher

Reta al usuario con operaciones sucesivas mientras un contador de tiempo se agota; se puede elegir entre 3 modos de dificultad en los cuales varia el tipo de operaciones y el tiempo para completarlas. Al finalizar este modo se muestran resultados del desempeño del jugador.

- Math Blaster Challenge

Un pequeño juego donde operaciones aparecen en la parte superior de la pantalla y descienden hacia el jugador; el usuario debe ingresar la respuesta correcta para evitar que las operaciones choquen contra él.

- Online World Challenge

Reta al jugador con una serie de operaciones; al finalizarlas muestra resultados del desempeño que obtuvo y lo compara contra jugadores alrededor del mundo.

2.5.2.2. Ventajas

- Varios modos de juego para poder practicar
- Modo online que permite retar a jugadores alrededor del mundo

2.5.2.3. Desventajas

- El modo Brain Cruncher es complicado de comprender cómo operarlo
- Anuncios intrusivos dañan la experiencia de juego
- La interfaz de usuario es poco atractiva

2.5.3. Kids Numbers And Math

Es un videojuego para dispositivos móviles Android diseñado para que niños de etapa primaria practiquen operaciones aritméticas elementales. En sus modos de juego ofrece las opciones de contar números, comparar números, sumar, restar y encontrar números iguales.

2.5.3.1. Funciones principales

- Contar

En este modo despliega una serie de objetos en pantalla y muestra cuatro números en la parte inferior de la pantalla; el usuario debe seleccionar el número que corresponde a la cantidad de objetos en pantalla.

- Máximos y mínimos

Muestra una serie de objetos en pantalla; cada uno con un número asignado; el usuario debe seleccionar el número mayor o menor según la aplicación lo requiere.

2.5.3.2. Ventajas

- Los ejercicios son acompañados de instrucciones audibles, esto hace más fácil comprender lo que se debe hacer y saber si la respuesta es correcta.

- Ofrecen ejercicios que pueden ser realizados por niños que están desarrollando sus habilidades aritméticas.

2.5.3.3. Desventajas

- Es limitado para usuarios de edad avanzada
- El audio solo está disponible en inglés
- Se debe de comprar una versión completa para acceder a todos los modos de juego

2.6. Solución tecnológica

Al analizar las diferentes opciones con que se cuenta para repasar aritmética en una forma que sea atractiva para los usuarios y que emplee la tecnología de dispositivos móviles; se ha decidido desarrollar un videojuego que combina la resolución de problemas matemáticos con los elementos de un juego de rol; dicho juego debe cumplir las siguientes características:

Tabla I. **Características deseadas en la solución tecnológica**

Facilidad de uso	El juego debe ser fácil de usar y amigable para el usuario; se debe contar con solo las opciones necesarias para jugar y no confundir al usuario; la interfaz debe ser limpia y agradable.
Operaciones aritméticas básicas	La aplicación debe tener soporte a las cuatro operaciones aritméticas básicas: suma, resta, multiplicación y división.
Progresión	El juego debe ir presentando retos de forma que el usuario perciba un incremento en la complejidad de las operaciones a realizar y una recompensa mayor al completarlas con éxito.
Practica de operaciones	El juego debe permitir al usuario practicar ciertas operaciones para reforzar el aprendizaje de operaciones específicas.
Localización de idiomas	La aplicación poseerá la opción de seleccionar diferentes lenguajes para que el idioma no sea un impedimento para el usuario.

Fuente: elaboración propia.

3. ARQUITECTURA DEL JUEGO

3.1. Sistema de entidad componente

Es un patrón de arquitectura utilizado generalmente en el desarrollo de juegos. Estos sistemas siguen el principio de composición sobre herencia, que permite más flexibilidad en definir entidades donde cada objeto en una escena de juego es una entidad. Cada entidad consiste en uno o más componentes, donde cada uno agrega un comportamiento o funcionalidad. Por lo tanto, el comportamiento de una entidad puede ser cambiado en tiempo de ejecución a través de la adición o eliminación de componentes.

Un sistema de entidad componente consiste en 3 partes primarias:

3.1.1. Componente

Un componente simplemente almacena cierto tipo de comportamiento o funcionalidad que le será adherida a la entidad que lo contenga.

3.1.2. Entidades

Una entidad es un objeto dentro del sistema que contiene una colección de componentes, los cuales definen su comportamiento y sus características.

3.1.3. Sistemas

Un sistema es usualmente una implementación que opera iterativamente sobre un grupo de entidades que comparten una colección específica de componentes.

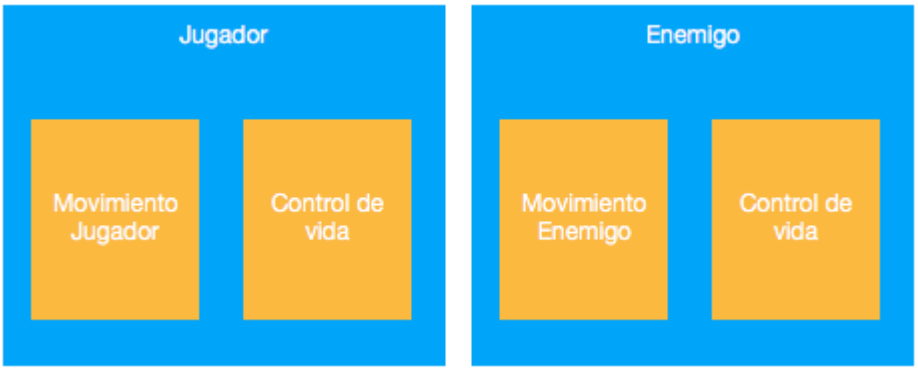
En el caso del motor de juegos de Unity, posee varios sistemas integrados, para facilitar el desarrollo de un juego, entre estos se encuentran:

- Sistema de partículas
- Sistema de representación de imágenes
- Sistema de física
- Sistema de iluminación

Para las entidades, Unity los define como objetos llamados objetos de juego, a los cuales se les puede manipular dentro del editor y agregar los componentes necesarios para su funcionamiento.

En el siguiente gráfico se ejemplifica como 2 entidades diferentes pueden tener componentes diferentes y, al mismo tiempo, ambos hacer uso del mismo componente.

Figura 7. **Sistema de entidades**



Fuente: elaboración propia.

3.2. Programación orientada a objetos

A pesar de que el manejo de entidades dentro del editor de Unity se realiza a través del sistema entidad componente, el sistema de programación, el cual está basado mayormente en el lenguaje de programación C#, sigue utilizando el paradigma de programación orientada a objetos.

Este paradigma está basado en el concepto de objetos, los cuales contienen datos, usualmente en la forma de campo, también conocidos como atributos, y código, en la forma de procedimientos, también conocidos como métodos.

Algunas de las características principales de este paradigma son:

3.2.1. Encapsulación

Es el concepto que une los datos y funciones que manipulan los datos y los mantiene a salvo de interferencia o mal uso fuera de la clase donde han sido creados.

3.2.2. Herencia

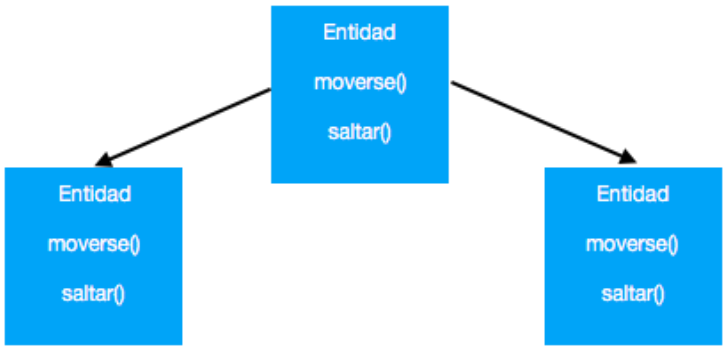
Este es el concepto que permite a las clases ser colocadas en una jerarquía que representa relaciones del tipo 'es un tipo de'. Esto permite la reutilización fácil de las mismas definiciones de procedimientos y datos, en adición a poder replicar relaciones del mundo real de una forma más intuitiva.

3.2.3. Polimorfismo

En un lenguaje de programación, esto es la provisión de una sola interfaz para diferentes tipos. En C# se utiliza usualmente el polimorfismo de inclusión, donde el nombre denota instancias de diferentes clases relacionadas por una superclase en común.

También, se puede dar el polimorfismo paramétrico el cual es escrito sin mención de ningún tipo en particular; por lo tanto, puede ser utilizado por varios nuevos tipos. Esto en programación orientada a objetos se les llama, muchas veces, genéricos.

Figura 8. **Orientado a objetos**



Fuente: elaboración propia.

4. DOCUMENTACIÓN BASE, DESARROLLO DE LA APLICACIÓN

El juego consiste en una serie de 'batallas matemáticas', las cuales son manejadas por medio de operaciones aritméticas básicas.

La pantalla de batalla está divida en 2 partes: en la parte inferior el usuario puede ingresar comandos usando un conjunto de botones similares a los de una calculadora para realizar las operaciones necesarias; la parte superior es donde se llevan a cabo la 'batalla', la cual se ve afectada de manera directa por las acciones del jugador mediante las operaciones que realice.

En cada batalla el jugador tendrá un rival al cual vencer; el rival es un avatar controlado por el CPU; para atacar deberá llenar su barra de poder; cuando la barra se llena el jugador puede atacar al rival lo cual sucede de forma automática con una animación.

Cada vez que el jugador responde correctamente una operación, cierto valor se suma a la barra de poder; este valor variará según el tiempo que le tome responder. Si el jugador responde de manera incorrecta, sumará a la barra de poder del rival. La barra del rival aumenta de forma automática según su nivel de dificultad.

La batalla termina cuando uno de los jugadores se queda sin energía para atacar.

4.1. Requerimientos de la aplicación

El juego surge de la necesidad de hacer la práctica de aritmética una tarea atractiva, mejorar la agilidad mental para realizar cálculos matemáticos y fomentar el gusto por las ciencias en los niños. Por lo que la aplicación debe contar con ciertas características, como:

- Modo de juego para un jugador
- Práctica de operaciones aritméticas
- Ser controlador con una pantalla táctil
- Línea gráfica 2D
- Ser intuitivo y fácil de utilizar
- Ser compatible con la mayoría de dispositivos móviles en el mercado

4.2. Tecnologías utilizadas

4.2.1. Unity Game Engine

Es un motor de juegos multiplataforma para desarrollar videojuegos 2D o 3D utilizando el lenguaje de programación C#. Está construido de forma que el editor juega una parte fundamental en el desarrollo; permite colocar entidades en una escena de juego de manera gráfica, estas entidades son extensibles utilizando código.

4.2.2. C#

Es un lenguaje de programación multiparadigma fuertemente tipado desarrollado por Microsoft para su plataforma .net. C# es un lenguaje diseñado por la infraestructura de lenguaje común (CLI).

4.2.3. Mono

Es una plataforma de código abierto basado en .NET; permite a los desarrolladores construir aplicaciones multiplataforma; Mono incluye un compilador de C#, un tiempo de ejecución y varias librerías. La implementación de .NET de Mono está basando en los estándares ECMA para C#.

4.2.4. IL2CPP

Es un sistema de *scripting* desarrollado por Unity como alternativa a Mono para construir proyectos en ciertas plataformas. Permite convertir código intermedio de *scripts* y ensamblados a código de C++ antes de crear un ejecutable para la plataforma destino. Esto permite obtener un mejor rendimiento, seguridad y compatibilidad.

4.3. Dependencias

4.3.1. Smart Localization

Es un plugin para el Unity Game Engine que provee un manejador de recursos de texto, audio e imágenes; permite asociar una llave a un conjunto de activos del juego, con el fin de tenerlos localizados en diferentes idiomas. Los archivos son almacenados en un documento de extensión rex y brinda la posibilidad de cambiar el idioma del juego sin tener que editar el código del juego.

4.3.2. Zestkit

Es una librería de *tweens* que permite crear animaciones desde código, de una forma concisa, fácil y potente. Una librería de *tweens* permite animar casi cualquier propiedad de un GameObject en el Unity Game Engine como cambiar su posición, escala y rotación o difuminar elementos de la interfaz de usuario modificando sus propiedades de colores.

Zestkit es el sucesor a GoKit y GoKit Lite, por lo que provee un marco de trabajo altamente modificable, eficiente y con un mínimo consumo de recursos.

La librería permite aplicar una función de aceleración a cada animación creada; esto permite crear efectos más llamativos que no solo inicien y se detengan abruptamente, sino que tengan ciertas variaciones a lo largo del tiempo en el que se ejecutan.

4.3.3. Arquitectura del juego

Para el desarrollo de la aplicación móvil se utilizó el patrón de diseño entidad componente y Singleton; Unity es un motor de juegos que se basa en entidad componente donde todo elemento es una entidad la cual puede contener una variedad de componentes que le permiten realizar diferentes acciones. Esto fomenta la reutilización de código.

4.3.4. Singleton

Se asegura que solo exista una instancia de una clase en un momento dado, provee un punto de acceso global al recurso.

4.4. Componentes del sistema

4.4.1. Generador de operaciones

El generador de operaciones se encarga de crear nuevas operaciones aritméticas utilizando un algoritmo que crea de forma aleatoria entre los cuatro posibles tipos de operación a realizar: suma, resta, multiplicación y división. Obtiene una operación válida y cuatro posibles soluciones, de las cuales solo una es la correcta.

Entre sus capacidades se encuentra:

- Generar una operación nueva
- Validar la respuesta del usuario

4.4.2. Control de energía

Lleva un control de los puntos de energía que una entidad necesita para poder realizar un ataque.

- Agregar puntos
- Restar puntos
- Notificar estado del jugador

4.4.3. Control de ataque

Se encarga del control del ataque del jugador o rival, este se incrementa cada vez que el jugador responde de forma correcta una operación.

- Añadir puntos
- Actualizar puntos

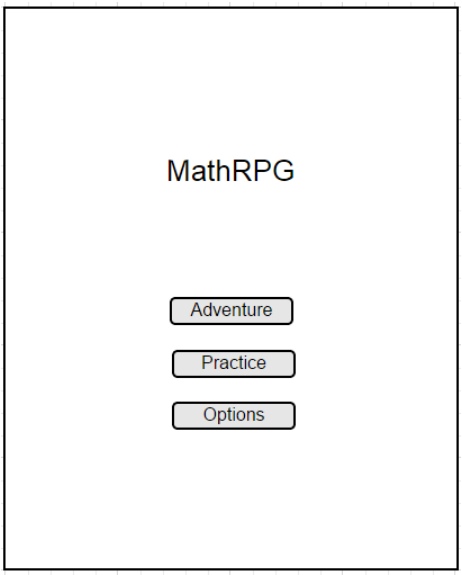
4.5. Diseño del prototipo

El prototipo muestra una prueba de concepto de la funcionalidad de los distintos módulos que conforman la aplicación.

4.5.1. Pantalla de inicio

Muestra el título del juego y permite al jugador acceder al menú de opciones para configurar el juego según sus necesidades o comenzar una partida nueva con la configuración por defecto o la última configuración guardada por el usuario.

Figura 9. **Pantalla de inicio**

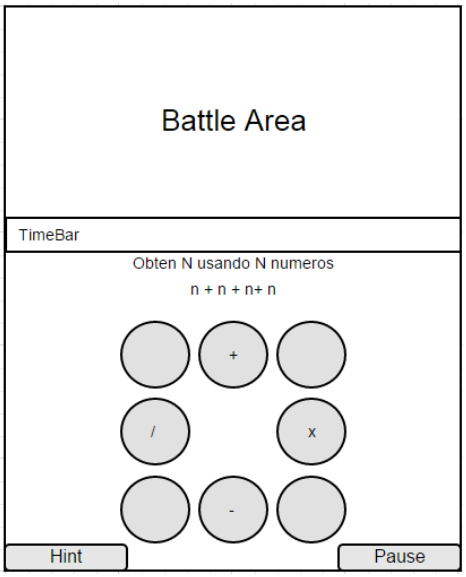


Fuente: elaboración propia.

4.5.2. Pantalla de juego

Muestra una partida en ejecución y se divide en dos partes: la parte superior de la pantalla muestra el área de juego con los avatares del jugador y el rival. En la parte media de la pantalla se muestra la barra de tiempo, la operación requerida y los operandos introducidos por el jugador. En la parte inferior de la pantalla se sitúan los botones de operandos y operadores y el botón de pausa.

Figura 10. **Pantalla de juego**

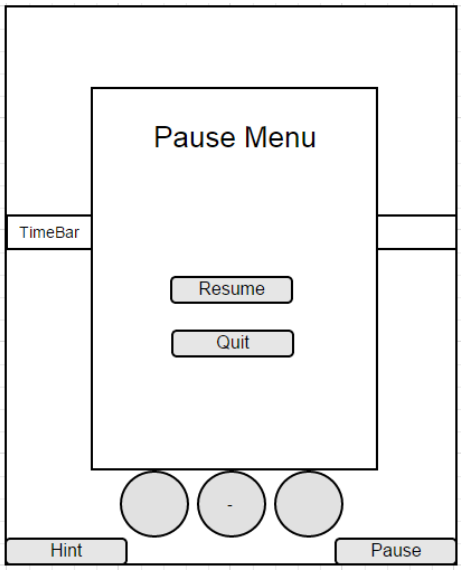


Fuente: elaboración propia.

4.5.3. Menú de pausa

Permite detener la ejecución del juego de forma momentánea; cuenta con las opciones de resumir el juego o salir al menú principal de la aplicación.

Figura 11. **Menú de pausa**



Fuente: elaboración propia.

4.6. Requisitos de uso de la aplicación

Para un funcionamiento óptimo del juego, se necesita como mínimo cumplir con los siguientes requerimientos de hardware:

- Pantalla táctil
- Chipset Qualcomm Snapdragon 400, Apple A5 o equivalente
- CPU Dual-Core 1GHz Cortex-A9
- GPU compatible con OpenGL 2.0 o superior
- 200 MB de espacio libre en memoria interna

- 512 MB de RAM

Los requerimientos mínimos de sistema operativo de los dispositivos móviles son los siguientes:

- Sistema operativo Android versión 4.0 o superior

- Sistema operativo iOS versión 7.1 o superior

5. DESARROLLO DE JUEGOS EN UNITY GAME ENGINE

Existe un conjunto de conceptos y elementos necesarios para desarrollar un videojuego en Unity.

5.1.1. Escena

Una escena es una estructura de datos en tiempo de ejecución del motor de juegos; permite alojar objetos en un espacio tridimensional; pueden ser utilizados para crear menús, niveles o posicionar cualquier ambiente, obstáculo y decoraciones.

5.1.2. Objetos de juego

Es el componente más importante del editor de Unity: todo en el juego es un objeto de juego. Los objetos de juego no tienen ninguna funcionalidad por sí solos, más que las propiedades para ocupar un lugar en el espacio; esto permite extenderlos dándoles propiedades para convertirse en cualquier cosa que el juego necesite.

5.1.3. Componente

Un componente es una pieza que permite a un objeto de juego poseer cierta funcionalidad, los componentes pueden ser puntos de emisión de luz, mallas de renderizado o incluso *scripts* de programación.

5.1.4. Transformada

Es un tipo de componente que permite determinar la posición, rotación y escala de un objeto en una escena. Todo objeto de juego tiene un componente transformado por defecto.

5.1.5. Cámara

Una cámara en Unity permite desplegar una escena a un jugador. Define una vista del espacio de la escena, la posición de la cámara en la escena define el punto de vista; los ejes hacia delante (Z) y hacia arriba (Y) definen la dirección de la vista y su orientación según la parte superior de la pantalla. Este componente también define el tamaño y la forma de la región que se ve en la pantalla.

5.1.6. Sitio de descarga de Unity Game Engine

La versión más reciente del Unity Game Engine se obtiene del siguiente enlace: https://unity3d.com/get-unity/download; después de instalarlo se podrá crear un nuevo proyecto.

5.1.7. Creación de un proyecto en Unity Game Engine

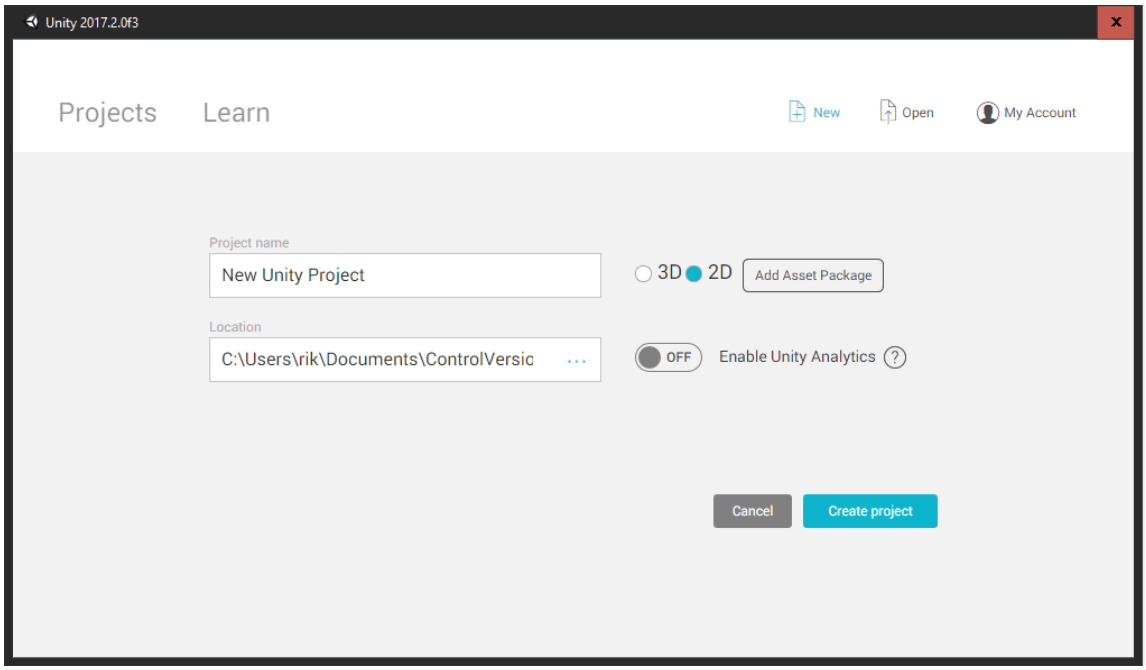
Se necesita crear una nueva aplicación de Unity; en la ventana de selección de proyectos se elige la opción New.

Figura 12. **Asistente de proyectos**

Fuente: elaboración propia.

Ahora, se procede a elegir el nombre del proyecto, donde será creado y que modo usará Unity por defecto, que permite elegir entre modo 2D y 3D. Para esta aplicación se utilizara 2D como modo predeterminado.

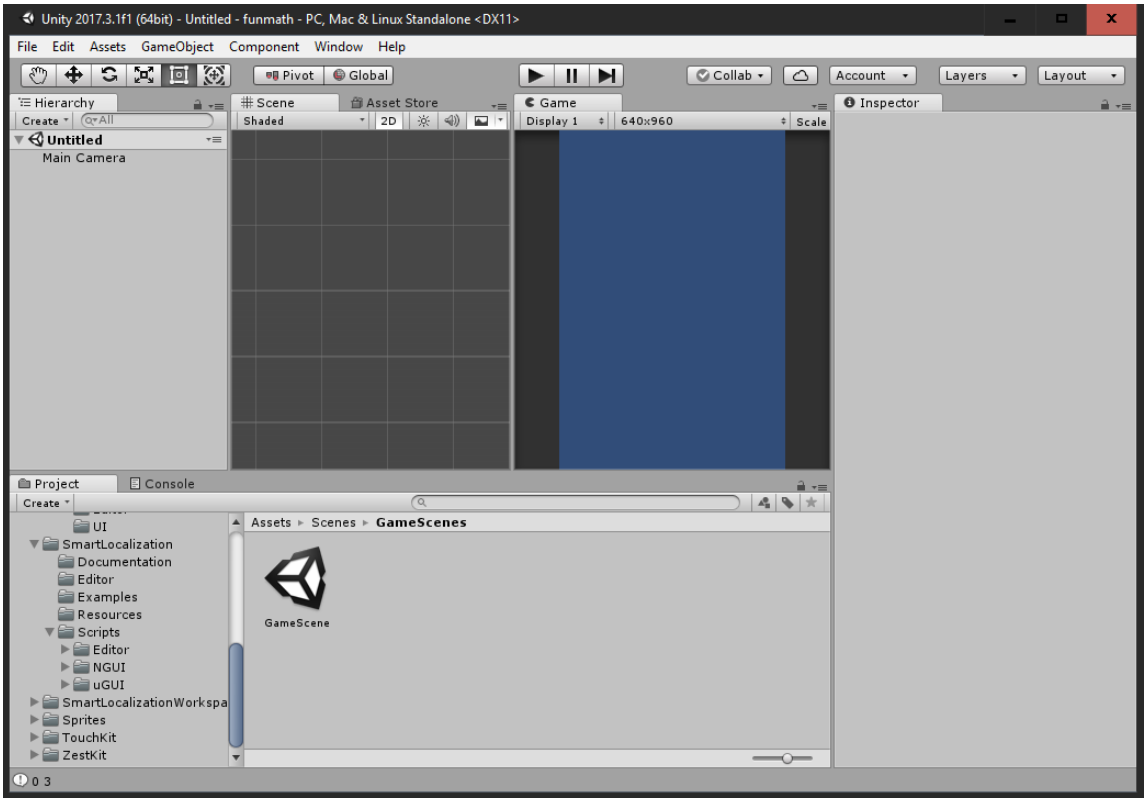
Figura 13. **Creación de proyecto nuevo**



Fuente: elaboración propia.

Esto creará un nuevo proyecto vacío, con una escena con una cámara con configuración por defecto en modo ortográfico para desarrollar un juego en dos dimensiones.

Figura 14. **Editor Unity en modo 2D**

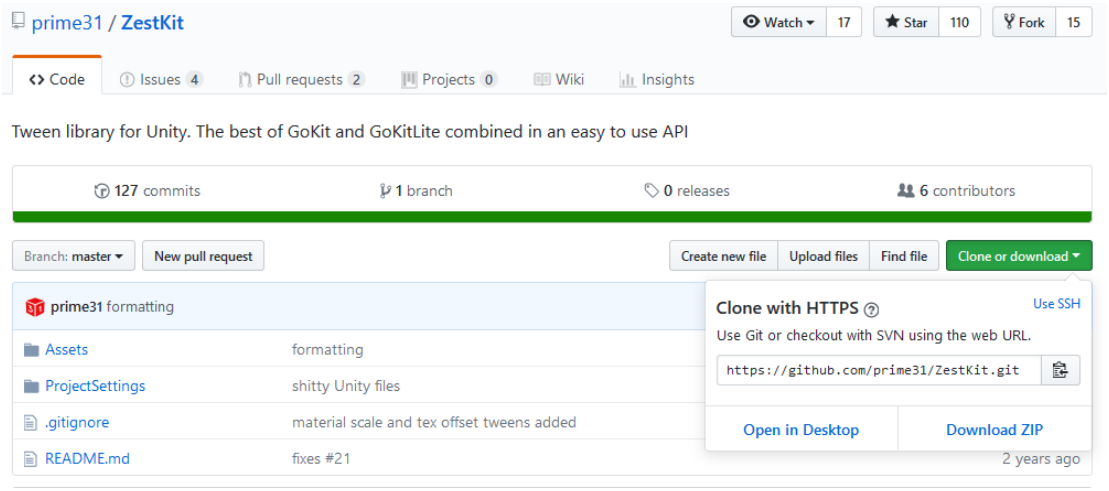


Fuente: elaboración propia.

5.1.8. Sitio de descarga de Zestkit

Zeskit es una librería de tweens para Unity Game Engine; se procede a descargarlo del siguiente vínculo: https://github.com/prime31/ZestKit; una vez descargado, se procede a colocar los archivos dentro de la carpeta Assets en el proyecto de Unity.

Figura 15. **Repositorio de ZestKit**

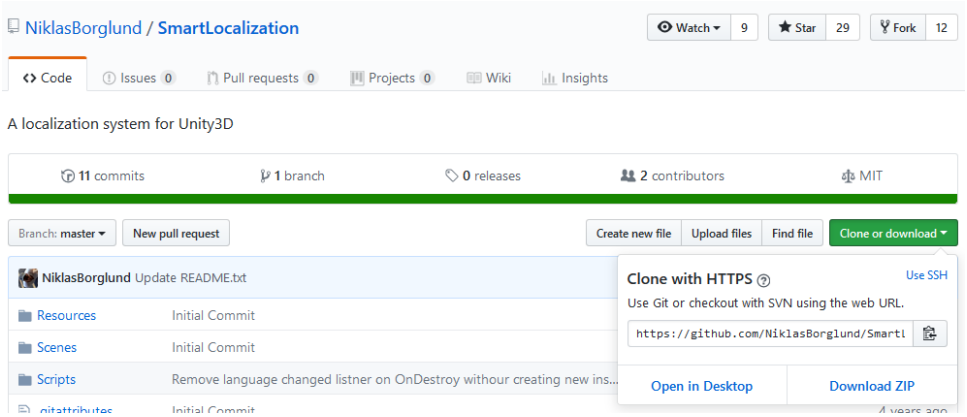


Fuente: *Github*. https://github.com/. Consulta: 29 de diciembre de 2017.

5.1.9. Sitio de descarga de Smart Localization

Smart Localization es una librería para manejar traducciones de bienes, puede descargarse del siguiente vínculo: https://github.com/NiklasBorglund/SmartLocalization; una vez descargado, se procede a colocar los archivos dentro de la carpeta Assets en el proyecto de Unity.

Figura 16. **Repositorio SmartLocalization**

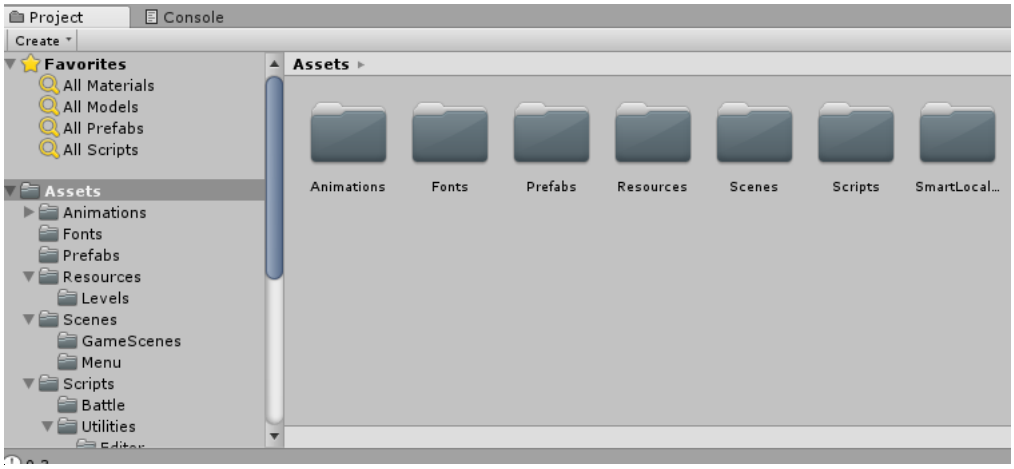


Fuente: *Github*. https://github.com/. Consulta: 29 de diciembre de 2017.

5.1.10. Estructura del proyecto

Se necesita crear un conjunto de directorios para ordenar los diferentes elementos del proyecto: scripts, prefabricados, imágenes y animaciones.

Figura 17. **Directorio del proyecto**



Fuente: elaboración propia.

5.1.10.1. Animations

En este directorio se almacenan todos los controladores de animación y sus referencias.

5.1.10.2. Scripts

Contiene los archivos con el código fuente de C# que brindan la lógica a cada componente del juego.

5.1.10.3. Scenes

Contiene todas las escenas del juego.

5.1.10.4. Prefabs

Contiene los prefabricados del juego, contenedores con componentes configurados y almacenados para su fácil instanciación cuando sea requerido.

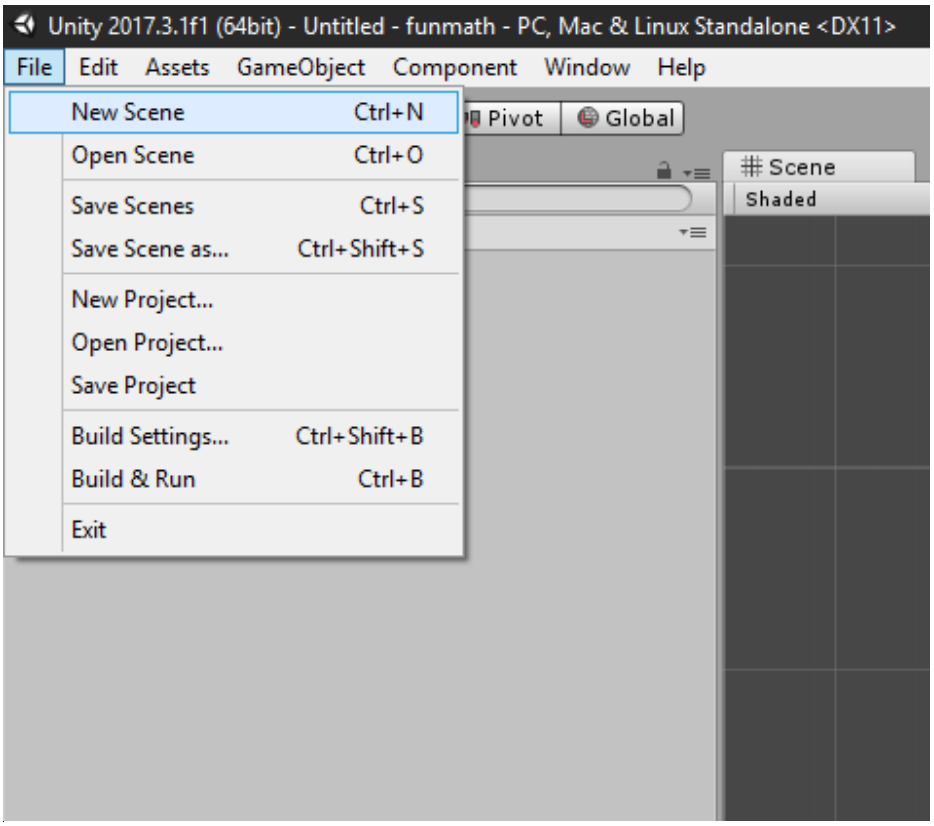
5.1.10.5. Recursos

Contiene los demás recursos del juego, en subdirectorios: sonidos, música, imágenes, etc.

5.1.11. Creando una escena de juego

Se necesita crear una escena que albergará todos los elementos del juego principal; en el menú de archivo; se selecciona la opción 'New Scene' o usando el atajo CRTL+N.

Figura 18. **Creación de una escena nueva**



Fuente: elaboración propia.

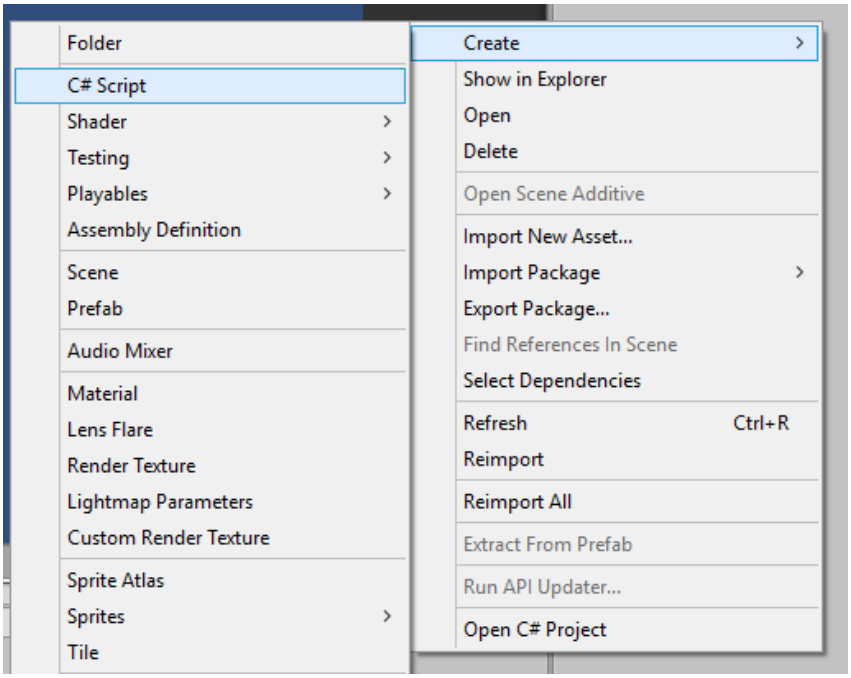
Se salva la escena de nombre GameScene dentro del directorio Scenes.

5.1.12. Creación de un Script de C#

El *scripting* permite agregar funcionalidad a los objetos de juego por medio de código; son componentes fundamentales en la creación de juegos en Unity por las posibilidades que brindan; un script puede controlar elementos en tiempo de ejecución del juego durante el ciclo de actualización; obtener componentes de otros objetos en las escenas y modificar sus propiedades y de esta manera reaccionar a los impulsos del usuario.

Para crear un *script* nuevo de C# basta con hacer clic derecho sobre el explorador de proyectos, elegir crear y seleccionar la opción C# Script.

Figura 19. **Creación de un *script***



Fuente: elaboración propia.

5.1.13. Detección de *input* táctil

Unity permite reconocer si el jugador ha tocado la pantalla táctil del dispositivo móvil donde la aplicación se ejecuta por medio del módulo de *Input*; cada toque en la pantalla es almacenado en una estructura llamada *touch*; esta estructura almacena la posición en pantalla y la fase; la fase hace referencia al estado en que un toque tiene en la pantalla; estos estados son: inicialización, finalización, movimiento, detenido, cancelado; al usarlos en conjunto, es posible crear un algoritmo capaz de reconocer gestos en pantalla.

5.1.13.1. Deslizar

Es un gesto que permite identificar si el usuario deslizó el dedo a través de la pantalla, el gesto es comúnmente utilizado por interfaces táctiles para mover objetos en pantalla, ocultar o mostrar objetos o ignorar alertas.

Con el gesto de deslizamiento, se puede obtener la dirección con la que el jugador ha deslizado el dedo y activar lógica por medio de un método que se llame al terminar el gesto. Para el menú de inicio del juego, se utilizó el siguiente código para detectar los deslices del jugador:

Figura 20. **Reconocimiento de deslices**

```
public class TouchResolutionScript : MonoBehaviour {
    bool move = false;
    TKSwipeRecognizer swipeRecognizer;
    Vector2 destination;

    void Start()
    {
        swipeRecognizer = new TKSwipeRecognizer();

        swipeRecognizer.gestureRecognizedEvent += (r) =>
        {
            switch (r.completedSwipeDirection.ToString())
            {
                case "Left":
                    destination = new Vector3(-Camera.main.pixelWidth, 0, 0);
                    move = true;
                    break;
                case "Right":
                    destination = new Vector3(Camera.main.pixelWidth, 0, 0);
                    move = true;
                    break;
                default:
                    destination = new Vector3(Camera.main.pixelWidth, 0, 0);
```

Fuente: elaboración propia.

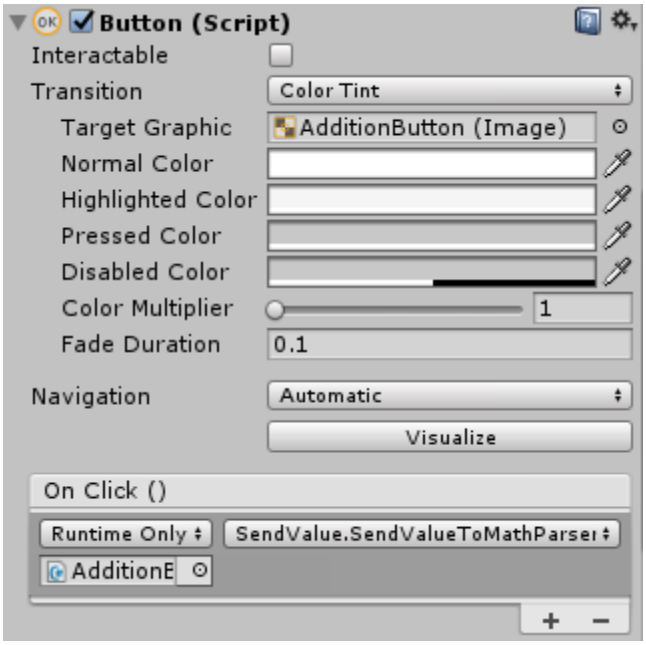
5.1.13.2. Interacción con elementos táctiles

Los elementos de interfaz de usuario de Unity permiten detectar la interacción con un puntero para hacerlo reaccionar como si fuese un botón es

posible agregar scripts que ejecuten código en cualquiera de los eventos del botón: al hacer clic, al soltar el clic o al cancelar el clic.

Se utiliza la siguiente configuración y el siguiente código para detectar cuando el jugador ha presionado uno de los botones del área de juego.

Figura 21. **Elemento UI con *script* de botón**



Fuente: elaboración propia.

Figura 22. **Código de *input***

```
public void SendValueToMathParser()
{
    if (Time.timeScale <= 0)
        return;

    GameLogic.AddToOperationStack(text.text);
    if (onlyOneUse)
    {
        GetComponent<Button>().interactable = false;
        GetComponent<Animator>().Play("Hide");
    }
}
```

Fuente: elaboración propia.

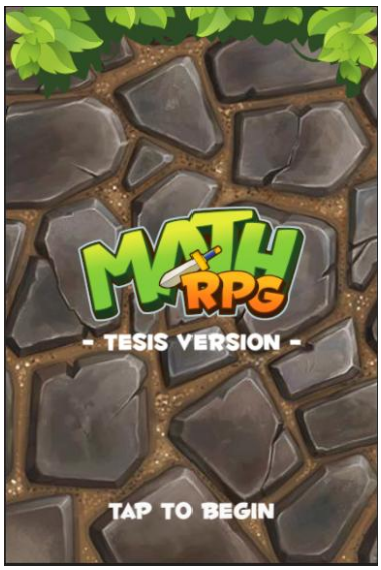
6. MANUAL DE USO DE LA APLICACIÓN

6.1. Pantalla de inicio

La pantalla inicial del juego recibe al jugador con una interfaz intuitiva basada en gestos táctiles simple de utilizar y perfecta para personas que están acostumbradas a interactuar con un dispositivo móvil o que utilizan uno por primera vez dada la simpleza de la interfaz de usuario.

Cuando un usuario inicia la aplicación, se despliega la pantalla con el título del juego y se le invita a tocar para iniciar; esto le permitirá acceder al menú principal del juego.

Figura 23. **Pantalla de inicio**



Fuente: elaboración propia.

6.2. Menú principal

El menú principal se activa cuando el jugador toca la pantalla; un contenedor se desliza en pantalla que ofrece al jugador dos opciones: iniciar una partida de practica nueva o personalizar las preferencias del juego.

Figura 24. **Menú principal**

Fuente: elaboración propia.

6.3. Pantalla de opciones

La pantalla permite modificar los parámetros de cada partida del juego; permite al jugador ajustar el juego a sus necesidades; estas opciones son persistentes en el dispositivo, permiten al usuario cerrar la aplicación y volver a iniciarla y conservar los ajustes realizados previamente.

6.4. El menú de opciones

6.4.1. Cambio del idioma del juego

El juego cuenta con opción para mostrar los textos en dos idiomas: inglés y español. El usuario puede elegir cualquiera de las dos opciones para jugar, por defecto la aplicación muestra todos los textos en inglés.

Figura 25. **Selección de idiomas**

Fuente: elaboración propia.

6.4.2. Cambio de los operandos disponibles en el juego

El usuario tiene la opción de elegir qué operaciones puede practicar dentro del juego: sumas, restas, multiplicaciones y divisiones. El juego generará operaciones basado en las preferencias de operaciones del jugador; por defecto, las cuatro operaciones se encuentran activas.

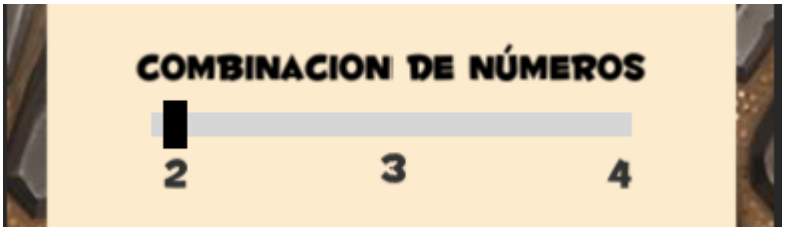
Figura 26. **Opciones de operandos**

Fuente: elaboración propia.

6.4.3. Combinación de números por operación

Permite que el usuario defina la complejidad de las operaciones; esto se hace con el número máximo de números a combinar por operación. El elemento deslizable permite seleccionar entre un mínimo de 2 operandos y un máximo de 4.

Figura 27. **Combinación de números**



Fuente: elaboración propia.

6.5. Pantalla de juego

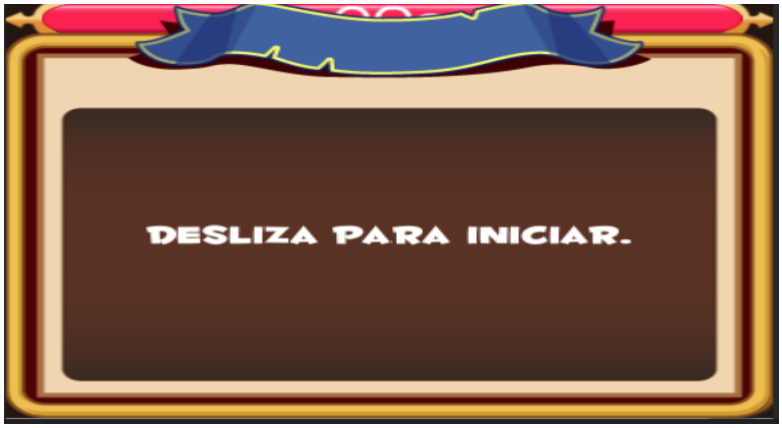
La pantalla de juego está dividida en dos áreas: la parte superior muestra a los avatares del juego, uno controlado por el jugador (del lado derecho) y otro controlado por el CPU (lado izquierdo).

En la parte inferior se encuentra el panel de interacción del usuario, con el cual puede iniciar el juego, responder las operaciones y acceder al menú de pausa.

6.5.1. Inicio de una partida nueva

Cuando el jugador inicia una partida nueva, el juego comienza hasta que realice un gesto de deslizar el dedo sobre la pantalla; de esta forma se desbloquea el panel numérico y el juego comienza.

Figura 28. **Iniciando una partida nueva**



Fuente: elaboración propia.

6.5.2. Resolución de operaciones

El juego genera una operación aleatoria según los operadores disponibles y el número de operandos combinables elegidos en el menú de opciones. Se le dan al jugador instrucciones que debe resolver antes de que el tiempo acabe.

El jugador debe responder de forma correcta utilizando el panel numérico e ingresando operandos y operadores.

Figura 29. **Área de operaciones**

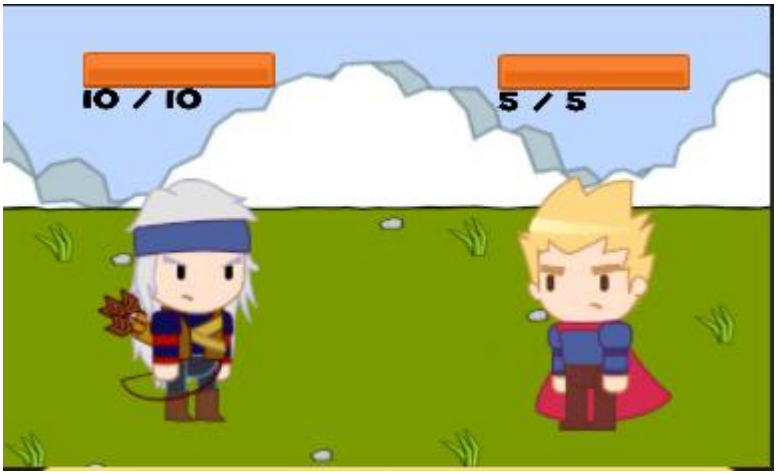
Fuente: elaboración propia.

6.6. Área de batalla

En la parte superior de la pantalla se lleva a cabo una batalla entre el jugador y un avatar controlado por el CPU; la barra de poder se llena cada vez que el jugador resuelve una operación exitosa y según la velocidad de su respuesta.

Los números debajo de la barra de poder de cada personaje muestra cuántos puntos de energía restantes tiene. Cuando el contador de energía se agota, el avatar pierde la batalla.

Figura 30. **Área de batalla**



Fuente: elaboración propia.

6.7. Menú de pausa

Este menú se activa al presionar el botón de pausa ubicado en el extremo inferior derecho de la pantalla. El juego se suspende momentáneamente

permitiendo al jugador resumir la partida usando el botón de continuar o regresar al menú principal.

Figura 31. **Menú de pausa**

Fuente: elaboración propia.

CONCLUSIONES

1. El uso de herramientas interactivas, como los juegos, puede ayudar a fomentar la participación de los estudiantes en materias científicas como las matemáticas.

2. El desarrollo de juegos y herramientas interactivas se ha vuelto mucho más accesible en los últimos años; permite a profesionales y estudiantes en países como Guatemala, adentrarse en la industria.

3. Herramientas como los juegos educativos ayudan no solo a fomentar el conocimiento en una materia específica; también, desarrollan otras habilidades de aprendizaje y memorización en los estudiantes mediante su uso.

RECOMENDACIONES

1. Las instituciones académicas, públicas y privadas, deben empezar a evaluar el uso de estas herramientas de aprendizaje alternativo para tener un mayor impacto en la educación de los estudiantes.

2. Las instituciones de educación superior orientadas a la tecnología deben empezar a integrar la enseñanza de desarrollo de juegos dentro de sus programas para mejorar la presencia nacional en esta área y permitir el desarrollo de más soluciones educativas y de entretenimiento.

3. Las entidades que manejan los programas educativos a nivel nacional pueden empezar a incentivar el uso de herramientas digitales, crear programas de apoyo para preparar a las futuras generaciones, en el ámbito laboral del futuro que se vuelve cada vez más digital.

BIBLIOGRAFÍA

1. Android. *API guides*. [en línea]. ‹http://developer.android.com/guide/index.html›. [Consulta: 20 de diciembre de 2017].

2. Apple. *Apple developer documentation*. [en línea]. ‹https://developer.apple.com/documentation›. [Consulta: 5 de octubre de 2017].

3. Common Sense Media Research. *Zero to eight: children's media use in America*. [en línea]. ‹https://www.commonsensemedia.org/research/zero-to-eight-childrens-media-use-in-america›. [Consulta: 3 de diciembre de 2017].

4. Khronos Group. *OpenGL reference page*. [en línea]. ‹https://www.khronos.org/registry/OpenGL-Refpages›. [Consulta: 15 de septiembre de 2017].

5. TUCKER, Allen. *The computer science handbook*. 2a ed. Estados Unidos de América: CRC PRESS, 2004. p. 350.

6. Unify Wiki. *Unify community Wiki*. [en línea]. ‹http://wiki.unity3d.com/index.php/Main_Page›. [Consulta: 10 de octubre de 2017].

7. Unity. *Script reference*. [en línea]. ‹https://docs.unity3d.com/ScriptReference›. [Consulta: 22 de diciembre de 2017].