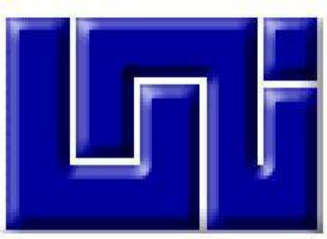
Universidad Nacional de Ingeniería

Recinto Universitario Simón Bolívar

Facultad de Electrotecnia y computación



TRABAJO MONOGRAFICO

VIDEO JUEGO EDUCATIVO EN 3D PARA DISPOSITIVOS MÓVILES ANDROID, PARA EL APRENDIZAJE DE LA LÓGICA DE PROGRAMACIÓN PARA LOS ESTUDIANTES DE LA CARRERA DE INGENIERÍA EN COMPUTACIÓN DE PRIMERO Y SEGUNDO AÑO.

Presentado por:

- Br. Isidro Antonio García Hernández **Carnet:** 2012- 41357
- Br. Manuel Adalberto Galán Huembes **Carnet:** 2012- 41340

Para Optar al Título de:

INGENIERO EN COMPUTACIÓN

Managua, Nicaragua Mayo del 2019

DEDICATORIA

Este trabajo monográfico constituye el producto de incansables días de sacrificios y entrega, pero felices y con la inmensa satisfacción por haber llegado a la cima de la montaña a la que la mayoría le teme, dedicamos humildemente nuestro trabajo:

A nuestro padre celestial que ilumina nuestros días y que lo sigue haciendo, que ha sido y seguirá siendo el inmenso motor que mueve nuestras vidas, a él infinitas gracias.

A nuestros maravillosos padres que siempre han estado ahí, apoyándome y dándome el gran amor que tienen para seguir adelante y nunca para atrás.

AGRADECIMIENTO

Primeramente, a Dios quien nos dio y nos da la vida, la sabiduría y las fuerzas para seguir luchando. A él sea la honra y la gloria**.**

A nuestros maestros de la facultad de Electrotécnica y Computación. Por su perseverancia y extraordinaria dedicación que empeño al permitirnos obtener sus conocimientos y vasta experiencia académica para el buen desarrollo de este trabajo Monográfico.

ÍNDICE GENERAL

1. INTRODUCCIÓN

El presente trabajo de tesis cuyo tema es, VIDEO JUEGO EDUCATIVO EN 3D PARA DISPOSITIVOS MÓVILES ANDROID, PARA EL APRENDIZAJE DE LA LÓGICA DE PROGRAMACIÓN PARA LOS ESTUDIANTES DE LA CARRERA DE INGENIERÍA EN COMPUTACIÓN DE PRIMERO Y SEGUNDO AÑO, consta de cinco capítulos que se detallan en forma organizada a continuación.

CAPÍTULO I: EL PROBLEMA, identifica el problema a investigar y además se plantea los antecedentes y la justificación por la cual se investiga, así como los objetivos a obtener los que guiarán la realización del proyecto.

CAPÍTULO II: MARCO TEÓRICO, presenta el fundamento teórico y los antecedentes investigativos que sustentan a la investigación y permiten comprender de manera clara el problema y así plantear la propuesta de solución.

CAPÍTULO III: METODOLOGÍA, describe la metodología de investigación a utilizar y el proceso de recolección, procesamiento y análisis de la información recolectada. Además, especifica de manera breve cada una de las etapas para el desarrollo del proyecto.

CAPÍTULO IV: DESARROLLO DE LA PROPUESTA, - en este capítulo se detalla de una manera clara el desarrollo de la propuesta, comparando diferentes frameworks y metodologías para desarrollo móvil que cumplan con los requerimientos del proyecto.

CAPÍTULO V: CONCLUSIONES Y RECOMENDACIONES, se establecen las conclusiones a las que se ha llegado luego del desarrollo del proyecto, así como recomendaciones que el investigador ha considerado pertinentes.

CAPÍTULO I

"**El Problema**", identifica el problema para resolver mediante un análisis previo, estableciendo en él sus **objetivos, antecedentes** y **justificación** y que llevaran a cabo la solución de una manera clara y concisa.

2. OBJETIVOS

2.1 Objetivo General:

- Desarrollar un videojuego educativo en 3D para dispositivos móviles Android, para el aprendizaje de la Lógica de Programación para los estudiantes de la carrera de Ingeniería en Computación de primero y segundo año.

2.2 Objetivos Específicos:

- ✓ Realizar un estudio de los principales Frameworks existentes en el mercado tecnológico para el desarrollo de video juego educativo para dispositivos móviles.
- ✓ Aplicar la metodología DAV (Desarrollo ágil para video juegos), para facilitar el desarrollo e implementación del proyecto planteado.
- ✓ Implementar el video juego educativo en 3D para dispositivos móviles, Android enfocado al aprendizaje de la Lógica de Programación.

3. ANTECEDENTES

Los videojuegos son juegos digitales interactivos, se trata de softwares ejecutables en dispositivos electrónicos diversos, tales como: computadoras, teléfonos móviles, consolas, tabletas entre otros. En general los videojuegos recrean entornos y situaciones en los que el jugador (o varios jugadores) pueden controlar personajes e interactuar con el entorno para alcanzar un objetivo determinado. Un juego puede resultar tanto o más enriquecedor, que cualquier otra actividad cuando el mismo ofrezca oportunidades de aprendizaje y sea abordado con la orientación, además de contar con las guías necesarias para aprovechar su valor educativo.

Realizando un estudio sobre los videojuegos móviles existentes en el mercado global, enfocados al aprendizaje de la lógica de programación se encontraron los siguientes:

MIT MEDIA LAB desarrolló un lenguaje de programación visual, diseñado para introducir las habilidades de programación en niños de 5-7 años de edad.

Mediante la creación de proyectos con ScratchJr, los niños pequeños pueden aprender a pensar de forma creativa y razonar de forma sistemática. Está disponible como una aplicación gratuita para iOS, Android y Chromebook. ScratchJr es un derivado del lenguaje Scratch, que ha sido utilizado por más de 10 millones de personas en todo el mundo. La codificación en Scratch requiere habilidades básicas de lectura, sin embargo, los creadores vieron una necesidad de otro lenguaje de programación, que proporcionaría una forma simplificada de codificación para aprender a una edad más joven y sin ningún tipo de lectura obligatoria[1].

Danny Yaroslavski realizó el videojuego educativo Lightbot para el aprendizaje de conceptos de programación de software. Este videojuego se ha jugado 7 millones

[1] Cbsnews.com, “Coding for kindergarteners: App teaches kids computer basics,” 2014. [Online] Available: https://goo.gl/C0KZhi

de veces, y es muy valorado en iTunes y Google Play Store, está disponible como un juego flash en línea, y una aplicación para teléfonos móviles Android y iOS. Lightbot se ha construido con Flash y OpenFL, el objetivo de Lightbot es mover a un pequeño robot para navegar por un laberinto usando diferentes comandos que se basan en conceptos de programación básicos y así poder encender unas luces y avanzar al siguiente nivel[2].

A nivel del país, no se han desarrollado videojuegos móviles enfocados al aprendizaje de la programación para universitarios de las carreras de Ingeniería en Computación e Ingeniería en Sistemas.

4. JUSTIFICACIÓN

Las necesidades cada vez más crecientes del sistema educativo, obligan a los estudiantes a ser parte de esta nueva era de cambio, en donde ya no solo deben dedicarse a estudiar, pasan a ser investigadores y empezar a innovar para llegar a ser competitivos.

La Programación sigue siendo uno de los pilares fundamentales para el desarrollo de la tecnología actual, además como se detalló en el planteamiento del problema, tanto las escuelas, como los colegios, no brindan la posibilidad del aprendizaje de la Lógica de Programación como una asignatura en su malla curricular, asignatura que actualmente es muy importante para el desarrollo del razonamiento lógico en los estudiantes de la carrera de Ingeniería en Computación de primero y segundo años.

Los estudiantes aprenden de mejor manera cuando se divierten, el modelo tradicional de aprendizaje muchas veces es aburrido, tedioso y mecánico, por lo que un videojuego, es la mejor forma de captar la atención y que el propio estudiante se convierta en el constructor de su aprendizaje.

[2] J. Biggs, “Light-Bot enseña ciencias de la computación con un pequeño robot lindo y un poco de programación basada en símbolos,” 2015. [Online] Available: https://goo.gl/IRPkmR

CAPÍTULO II

"**Marco Teórico**", consta de los fundamentos teóricos que serán base para comprender de manera adecuada y precisa del problema planteado, además será un apoyo científico que guiará durante el desarrollo del proyecto.

Para la total compresión de esta propuesta de protocolo monográfico es necesario estudiar los conceptos elementales de los videos juegos en 3D, las técnicas y procedimientos empleados en un caso.

4.1 Concepto de Video juego

Dentro del mundo del entretenimiento electrónico, un videojuego normalmente se suele asociar a la evolución, entendida desde un punto de vista general, de uno o varios personajes principales o entidades, que pretenden alcanzar una serie de objetivos en un mundo acotado, los cuales están controlados por el propio usuario[3].

De este modo, existe una interacción explícita, entre el jugador o usuario de videojuegos y el propio videojuego, el cual plantea una serie de retos al usuario con el objetivo final de garantizar la diversión y el entretenimiento. Además de ofrecer este componente emocional, los videojuegos también suelen tener un componente cognitivo asociado, obligando a los jugadores a aprender técnicas y a dominar el comportamiento del personaje que manejan para resolver los retos o puzzles que los videojuegos plantea[3].

4.2 Video juegos “Indies”

Los video juegos “Indies”, nacieron como una filosofía (o necesidad) de crear videojuegos en una habitación con poco o ningún presupuesto, 2 o 3 personas dejando su alma para crear un video juego, sin seguir las normas impuestas de los productores.

Los juegos independientes empezaron como una corriente alternativa, en la que la originalidad no iba de la mano con el éxito. Apuestas arriesgadas que no tenían por qué ser populares y que descubrieron una inquietud en muchos de los jugadores empachados de súper producciones. Los productores no se arriesgan,

[3] D. V. Fernandez, C. M. Angelina, and EspaCursos, Desarrollo de Videojuegos: Arquitectura del Motor de Videojuegos. Cursos en Español, Oct. 2011

no deja de ser un negocio y lo que quieren, es ganar dinero de forma segura (o todo lo seguro que puede ser este mercado)[4].

Crear un video juego nunca había sido tan fácil (desde la época de los 8 bits con Basic como estandarte) ya que existen cientos de herramientas que reducen el duro camino del desarrollo. Programas como Game Maker o Unity 3D permiten hacer cosas increíbles con un menor esfuerzo. Además, con la era de la información en la que se vive, es fácil aprender cualquier cosa por medio del internet [4].

4.3 Video juegos educativos

En los últimos años aumentó sensiblemente la oferta de videojuegos educativos, que se presentan como una alternativa a los videojuegos tradicionales. Este incremento viene motivado por varios factores, como se extrae de los estudios de Pérez Martín, entre los que destaca la madurez de las empresas desarrolladoras españolas, lo que implica productos de gran calidad con buenos guiones y acabados, entrada en la cadena de distribución y actividades de promoción con el fin de ser conocidos por el público[5].

Los video juegos favorecen los reflejos, la psicomotricidad, la iniciativa y la autonomía, pudiéndose introducir en la educación con una finalidad didáctica, para contribuir al logro de determinados objetivos educativos. Existen diferentes aspectos que permiten desarrollar los videojuegos como son:

1. Aspectos cognitivos: memorización de hechos, observación hacia los detalles, percepción y reconocimiento espacial, descubrimiento inductivo, capacidades lógicas y de razonamiento, comprensión lectora y vocabulario, conocimientos geográficos, históricos, matemáticos, resolución de problemas y planificación de estrategias[5].

[4] Pixelsmil.com, “Qué significa "Indie",” 2012. [Online] Available: http://www.pixelsmil.com/2012/05/que-significa-indie.html.
[5] . P. Martín, “Penetración de los videojuegos educativos e infantiles en España desde el 2005 al 2007,” Comunicación y Pedagogía: Nuevas tecnologías y recursos didácticos, no. 229, pp. 23–28, 2008.

2. Destrezas y habilidades: autocontrol y autoevaluación, implicación y motivación, instinto de superación, inversión de esfuerzo que es reconocido de forma inmediata, habilidades motrices, de reflejos y respuestas rápidas, percepción visual, coordinación óculo-manual, y percepción espacial, curiosidad e inquietud por probar y por investigar[5].

3. Aspectos socializadores: aumenta la autoestima, proporcionan un sentido de dominio, control y cumplimiento, debido en gran parte a que existen recompensas personalizadas, interacción con amigos de manera no jerárquica (presencial o a distancia) [5].

4.4 Motor de juego

El término motor de juego surgió a mediados de los años 90, con la aparición del famoso juego de acción Doom, desarrollado por la compañía id Software bajo la dirección de John Carmack.

4.5 Arquitectura de un motor de juego

Como ocurre con la gran mayoría de sistemas software que tienen una complejidad elevada, los motores de juegos se basan en una arquitectura estructurada en capas. De este modo, las capas de nivel superior dependen de las capas de nivel inferior, pero no de manera inversa.

Gestor de recursos. - Esta capa es la responsable de proporcionar una interfaz unificada para acceder a las distintas entidades software que conforman el motor de juegos, como por ejemplo la escena o los propios objetos 3D[3].

Motor de rende ring. - El motor de renderizado es una de las partes más complejas de cualquier motor de juego. En la capa de rendering sólo se dibujan las primitivas que están dentro del campo de visión de la cámara, es decir, dentro del viewport, es posible aplicar más optimizaciones que simplifiquen la complejidad de la escena a renderizar, obviando aquellas partes de la misma que no son visibles desde la cámara.

Motor de física. - La detección de colisiones en un videojuego y su posterior tratamiento, resultan esenciales para dotar de realismo al mismo. Sin un mecanismo de detección de colisiones, los objetos se traspasarían unos a otros y no sería posible interactuar con ellos. Un ejemplo típico de colisión está representado en los juegos de conducción por el choque entre dos o más vehículos [3].

Interfaces de usuario. - En cualquier tipo de juego es necesario desarrollar un módulo que ofrezca una abstracción respecto a la interacción del usuario, es decir, un módulo que principalmente sea responsable de procesar los eventos de entrada del usuario. Típicamente, dichos eventos estarán asociados a la pulsación de una tecla, al movimiento del ratón o al uso de un joystick, entre otros.

Audio. - Tradicionalmente, el mundo del desarrollo de videojuegos siempre ha prestado más atención al componente gráfico. Sin embargo, el apartado sonoro también tiene una gran importancia para conseguir una inmersión total del usuario en el juego. Por ello, el motor de audio ha ido cobrando más y más relevancia.

4.6 Metodologías tradicionales de desarrollo de software

Las metodologías tradicionales son denominadas, a veces, de forma despectiva, como metodologías pasadas, centran su atención en llevar una documentación exhaustiva de todo el proyecto y en cumplir con un plan de proyecto, definido todo esto, en la fase inicial del desarrollo del proyecto. Otra de las características importantes dentro de este enfoque, son los altos costes al implementar un cambio y la falta de flexibilidad en proyectos donde el entorno es volátil. Las metodologías tradicionales (formales) se focalizan en la documentación, planificación y procesos (plantillas, técnicas de administración, revisiones, etc.)[6].

Entre las principales metodologías tradicionales tenemos los ya conocidos RUP-RATIONAL UNIFIED PROCESS y MSF (MICROSOFT SOLUTION FRAMEWORK) entre otros, que centran su atención en llevar una documentación

[6] L. N. de Calidad del Software (España), “Metodologías y Ciclos de Vida,” 2009. [Online] Available: https://goo.gl/bkO7yO.

exhaustiva de todo el proyecto y en cumplir con un plan de proyecto, definido todo esto, en la fase inicial del desarrollo del proyecto.

RUP. - Es un proceso formal: Provee un acercamiento disciplinado para asignar tareas y responsabilidades dentro de una organización de desarrollo. Su objetivo es asegurar la producción de software de alta calidad que satisfaga los requerimientos de los usuarios finales (respetando cronograma y presupuesto). Fue desarrollado por Rational Software, está integrado con toda la suite Rational de herramientas. Puede ser adaptado y extendido para satisfacer las necesidades de la organización que lo adopte. Es guiado por casos de uso y centrado en la arquitectura, y utiliza UML como lenguaje de notación[7].

MSF. - Es un enfoque personalizable para entregar con éxito soluciones tecnológicas de manera más rápida, con menos recursos humanos y menos riesgos, pero con resultados de más calidad. MSF ayuda a los equipos a enfrentarse directamente a las causas más habituales de fracaso de los proyectos tecnológicos y mejorar así las tasas de éxito, la calidad de las soluciones, además el impacto comercial[8].

4.7 Metodologías ágiles

El desarrollo ágil de software, se refiere a métodos de Ingeniería del Software basados en el desarrollo iterativo e incremental, estas metodologías son imprescindibles en un mundo en el que se expone a cambios recurrentemente. Siempre hay que tener en cuenta en la programación la última tendencia de hoy, puede que no exista mañana y por esto existe la metodología ágil donde los requisitos y soluciones evolucionan mediante la colaboración de grupos auto organizados y multidisciplinarios[9].

[7] P. Kruchten, El proceso racional unificado: una introducción. Addison-Wesley Professional, 2011.

[8] Msdn.microsoft, “Microsoft Solutions Framework (MSF),” 2016. [Online] Available: https://msdn.microsoft.com/es-es/library/jj161047(v=vs.120).aspx.

[9] O. Pastrana, “5 beneficios de aplicar metodologías ágiles en el desarrollo de software,” 2014. [Online] Available: https://goo.gl/wCMAvt.

Según el estudio de la Universidad de la República de Uruguay, las empresas que realizan video juegos en equipos pequeños y de corta duración no tienen una metodología de desarrollo formalizada. Las metodologías que utilizan siguen principios de las metodologías ágiles, que se adaptan con éxito para el desarrollo de videojuegos a nivel mundial y aplican a realidades similares. En particular se registran casos de éxito con adaptaciones de SCRUM y XP (Extreme Programming), aunque estas tampoco se encuentran formalizadas[10].

SCRUM. - El SCRUM es un proceso de la Metodología Ágil que se usa para minimizar los riesgos durante la realización de un proyecto, no obstante, de manera colaborativa. Entre las ventajas se encuentran la productividad, calidad y que se realiza un seguimiento diario de los avances del proyecto, logrando que los integrantes estén unidos, comunicados y que el cliente vaya viendo los avances. La profundidad de las tareas que se asignan en SCRUM tiende a ser incremental, caso que coincide exactamente con el devenir normal de un desarrollo.

XP. - La "programación extrema" es un proceso de la Metodología Ágil que se aplica en equipos con muy pocos programadores quienes llevan muy pocos procesos en paralelo. Consiste entonces en diseñar, implementar y programar lo más rápido posible, hasta en casos se recomienda saltar la documentación y los procedimientos tradicionales. Se fundamenta en la capacidad del equipo para comunicarse entre sí y las ganas de aprender de los errores propios inherentes en un programador.

La tendencia a utilizar metodologías ágiles para videojuegos tomó fuerza en los últimos años, por existir varios casos de empresas en la industria que logran adaptar estas metodologías, entre las empresas con casos de éxito documentados se encuentran Crytek y DICE (EA Digital Illusions Creative Entertainment) que utilizan SCRUM, Titus Interactive Studios que utiliza XP y High Moon Studios que

[10] N. Acerenza, A. Coppes, G. Mesa, A. Viera, E. Fernández, T. Laurenzo, and D. Vallespir, "Una Metodología para Desarrollo de Video juegos," 2009.

utiliza ambas. A pesar de los beneficios que reportan, ninguna de estas adaptaciones está especificada formal y públicamente.

4.8 Lógica de programación

Consideraciones algorítmicas sobre el pensamiento humano Los algoritmos informales. Son aquellos que no se realizan para ser ejecutados por una computadora, sino se diseñan para ser ejecutados por el ser humano. Todos los días se ejecutan algoritmos informales para realizar actividades: al prepararse para ir al trabajo, al vestirse, al comer, al regresar a casa, etc.

Los algoritmos computacionales. Son aquellos que son ejecutados por una computadora. Se aprovecha la velocidad de procesamiento del ordenador para obtener un resultado mucho más confiable, por ejemplo, el cálculo de una raíz cuadrada.

Analizando desde muchos ángulos del pensamiento humano y teniendo en cuenta los conceptos de algoritmo informal y algoritmo computacional, se llegó a la conclusión de que dicho pensamiento se mueve entre tres estructuras básica:

1. Secuencias
2. Decisiones
3. Ciclos

-Secuencias

Para escribir una secuencia de ordenes o acciones todo lo que tiene que hacer es colocar una nueva orden o una nueva acción después de la última que haya colocado. De esta manera se entiende la secuencialidad y la ordinalidad en la ejecución de esas acciones.

El siguiente algoritmo permite asomarse por la ventana, no obstante, asumiendo que la ventana ya está abierta y que no existe mucha distancia hacia la ventana. Por lo tanto, el algoritmo quedaría de la siguiente manera:

1 Algoritmo para asomarnos a la ventana

2 I n i c i o

3 Ubicar la ventana por la que nos queremos asomar

4 Levantarnos del lugar en donde estemos sentados

5 Avanzar hacia la ventana llegar hasta tener la ventana muy muy cerquita

6 Asomarnos por la ventana

7 Fin

En el ejemplo dado, se puede ver que cada acción, está antes de una y después de otra (excepto por supuesto la primera y la última). También puede notar que para que este algoritmo permite asomarse por la ventana, todo lo hay que hacer, es realizar cada acción en el orden en que están planteadas y sencillamente realizar una a la vez, eso permite lograr el objetivo propuesto.

-Decisiones

Siempre que se necesite tomar una decisión o, más bien, siempre que se utilice la estructura de Decisiones será necesaria una condición. La condición es lo que permite elegir que camino lógico tomar.

A continuación, se muestra el mismo algoritmo de asomarse a la ventana, no obstante, esta vez se añadirán unas líneas de decisión, que permitirán crear la estructura de decisiones, así:

1 Algoritmo para asomarnos a la ventana

2 I n i c i o

3 Ubicar la ventana por la que nos queremos asomar

4 Si estamos sentados

5 Levantarnos del lugar en donde estemos sentados

6 Orientarnos hacia la ventana

7 Sino

8 Orientarnos hacia la ventana

9 Avanzar hacia la ventana

10 Llegar hasta tener la ventana muy muy cerquita

11 Si está cerrada Abrirla

12 Asomarnos por la ventana

13 Fin

1.Las palabras **Si** que aparecen son exclusivamente condicionales y no afirmativas como pudiera pensarse en algunos casos.

2.Después de cada **Si** condicional, va una condición, que es la que permite que se haga una cosa u otra. La condición regula las acciones que vienen después y que dependen del **Si** condicional inicial. En la decisión.

-Ciclos

Se va a suponer que, un supervisor de una fábrica a lo largo de todo el día, debe estar vigilando determinada acción a través de una ventana. El algoritmo para cumplir su objetivo que es el de Vigilar (como supervisor de la fábrica) parte de una tarea muy sencilla la cual es “asomarse” por una ventana. En palabras sencillas el supervisor tendrá, que asomarse por una ventana mientras no termine el día. De esta forma, la estructura para este algoritmo es la siguiente:

1 Algoritmo para Vigilar desde una ventana

2 Inicio

3 Llegar puntual a la hora de inicio de la jornada laboral

4 Ubicarnos en nuestro escritorio

5 Sentarnos cerca de la ventana

6 Mientras no sea e l fin del día

7 Asomarse por la ventana

8 Fin_Mientras

9 Fin

***Varios factores nuevos entran en este algoritmo:**

1. La palabra Mientras establece en relación con una condición el inicio de un conjunto de acciones, que se repiten precisamente mientras esa condición lo permita.
2. Todo Mientras (por efectos de clarificación del algoritmo) debe tener un finalizador, que indique hasta donde llega el bloque de acciones que deben repetir.
3. La indentación o lo que corrientemente se conoce como el “Sangrado” del texto es decir el hecho de que algunas acciones, estén más adentro de la hoja que otras, representa que existen bloques de acciones que tienen una característica.

    - Las acciones contenidas entre el Inicio y el Fin, indican que son las acciones que conforman el algoritmo en mención.
    - Las acciones comprendidas entre Mientras no sea Fin del día y su correspondiente Fin_Mientras, son el conjunto o bloque que se debe repetir (o iterar) precisamente mientras la condición sea Verdadera o sea Mientras no sea fin del día.

4. Cada ciclo de acciones que se inicie con Mientras, deberá tener un Fin_Mientras asociado y a su vez, cada Fin_Mientras deberá finalizar uno y solo un ciclo iniciado con Mientras.

CAPÍTULO III

"**Metodología**", se indica la metodología que se utilizaran especificando además las técnicas e instrumentos para recolectar y procesar la información, también describe el camino que deberá seguir para el desarrollo del proyecto.

5. DISEÑO METODOLÓGICO

5.1 Metodología de desarrollo del proyecto

DAV (Desarrollo Ágil de Videojuegos), es una metodología que nace con la unión de las características de SCRUM y XGD, con la finalidad de proporcionar una estructura que permita la creación de videojuegos de forma técnica y sencilla. Esta metodología se compone de los siguientes elementos: valores, prácticas, roles, reuniones, fases, artefactos y herramientas.

***Prácticas**

Esta metodología utiliza prácticas tanto de SCRUM como de XGD para integrar en la nueva metodología DAV, a continuación, se describe cada una de ellas:

- **Diseño incremental:** Las tareas de un videojuego deben ser de la forma más sencilla posible, que funcionen correctamente.
- **Historias de usuario:** Se trata de una breve descripción de las funcionalidades del videojuego, por lo general descrito por el Dueño del Producto.
- **Ciclos semanales:** El proyecto está organizado y planificado para ser ejecutado en ciclos de corta duración.
- **Entregas pequeñas:** Producir rápidamente versiones del videojuego estables, aunque no cuenten con toda la funcionalidad del videojuego. Esta versión ya constituirá un resultado de valor para el negocio.
- **Pruebas:** El videojuego será probado en varios momentos por diferentes personas, dando como resultado una información valiosa que podrá ser tomada como base para decisiones futuras.

***Roles**

Los roles de DAV se basan únicamente en SCRUM, añadiendo al equipo DAV el rol de tester, siendo estos necesarios y suficientes para ser aplicados en equipos pequeños, clasificándose en dos tipos de roles:

- **Dueño del Producto:** El Dueño del Producto es la única persona autorizada para decidir sobre cuáles funcionalidades y características funcionales tendrá el videojuego. Es quien representa al cliente y todas aquellas partes interesadas en el videojuego, además es quien maneja y prioriza las funcionalidades a desarrollar y las coloca en la Reserva de Producto.
- **DAV Máster:** El DAV Máster no es un líder típico, sino un auténtico servidor neutral, que será el encargado de enseñar la metodología DAV a cada integrante implicado en el proyecto, preocupándose de poner la metodología en práctica de modo que se encuentre dentro de la cultura de la organización y así entregue las ventajas previstas, asegurándose de que cada uno siga las reglas y prácticas de DAV.
- **Equipo DAV:** Es un equipo multidisciplinario y auto-organizado, su función principal es construir el videojuego que el Dueño del Producto específico. Los miembros del equipo pueden ser de 3 a 9 personas, entre los cuales tenemos:
    - **Diseñador:** Realiza el diseño conceptual, define las reglas y la mecánica, escribe el guión y diseña los niveles.
    - **Programador:** Implementa la funcionalidad pedida en el diseño, programa la lógica del juego, gráficos (efectos), inteligencia artificial y audio y conoce de herramientas/motores (infraestructura).
    - **Artista:** Construye los objetos, personajes, niveles y anima los personajes.
    - **Sonidista:** Crea los sonidos, estilos, efectos y ambientes sonoros para el videojuego.
    - **Tester:** Verifican que el contenido, funcionalidad del videojuego son correctos, completos y evalúan la jugabilidad.
- **Interesados (Clientes):** Se refiere a la gente que hace posible el proyecto y para quienes el videojuego producirá el beneficios acordado que justifica su producción. Sólo participan directamente durante las revisiones de Iteración.

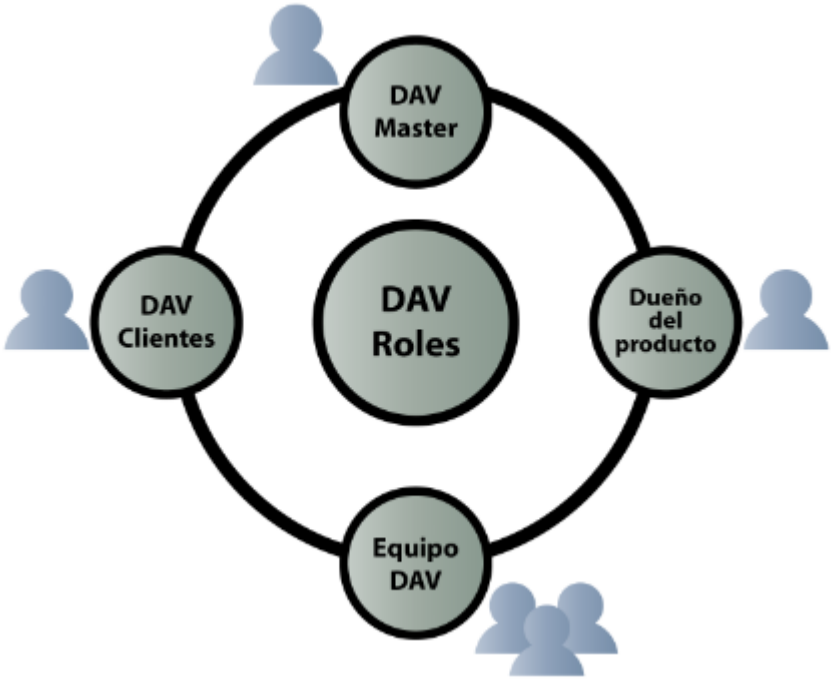


Figura #1: Roles del DAV

Fuente: Isidro García y Manuel Galán

***Fases**

La división de trabajo en el desarrollo de videojuegos no se distribuye por igual en sus iteraciones; para apoyar esta realidad DAV se basa únicamente en las fases de SCRUM: pre-juego, juego y post-juego.

1. **Fase pre-juego**

En esta fase se realiza la concepción de la idea del juego del dueño del producto con ayuda de los miembros del equipo, es decir, los aspectos fundamentales que conformarán el videojuego:

- **Género:** Se debe especificar el género o géneros al que pertenece el videojuego para establecer las características básicas que tendrá en su posterior diseño.
- **Historia:** Se debe realizar un bosquejo de la trama o historia del videojuego a desarrollar, indicando qué se quiere contar y cómo se quiere contar.

- **Bocetos:** Se crean bocetos o diseños preliminares de los personajes y en dónde trascurrirá la acción del videojuego, ya sean decorados, ambientaciones, ropaje, música, movimientos, etc.
- **Aspecto:** A partir de los bocetos define el aspecto gráfico y artístico del videojuego, colores, temas dominantes, musicalidad, técnicas de diseño 3D o 2D, posiciones de cámaras, etc.
- **Interfaz de usuario:** Manera de cómo interactuará el jugador con el videojuego y con qué mecanismos contará para ello.
- **Objetivos:** Cuáles son las metas del videojuego, de acuerdo a la historia de éste.
- **Reglas:** Qué cosas podemos hacer y cómo vamos a dejar que se hagan.
- **Características:** Especificación de las principales características de cada personaje del videojuego y de los elementos que intervienen en éste.
- **Jugabilidad:** Es aquí donde se definirá cómo se va a jugar, de qué manera se va a jugar, qué cosas podemos hacer en el juego y cómo va reaccionar el entorno del juego a las acciones del jugador a través del personaje. A su vez se debe establecer cómo será la curva de aprendizaje del jugador. Todo esto sin entrar en detalles gráficos, sonoros o de historia.
- **Diseño de niveles:** Describir qué niveles, según la historia, se tendrá, cómo serán éstos, cuántos serán, y qué retos se plantearán en cada uno de ellos.
- **Requerimientos técnicos:** Establecer los requerimientos técnicos del equipo que necesitará el videojuego para poder ejecutarse.
- **Marketing:** Parte fundamental, debido a que muchos videojuegos de inversiones millonarias han ido al traste por una mala campaña de publicidad, para evitar esto hay que establecer las líneas de publicidad para el videojuego.

La fase pre-juego reduce la necesidad de encontrar ese elemento elusivo de diversión durante la etapa de juego y permite al equipo centrarse en la ejecución del videojuego, en lugar de experimentar con él.

2. Fase juego

En esta fase se tiene bien definido el alcance del proyecto, por lo tanto, ya existe una idea clara de lo que realmente es el videojuego y lo que se debe hacer.

- **Iteraciones:** Esta fase que suele llamarse iteración, consta de ciclos que pueden durar de 1 a 4 semanas (por lo general 1-2 semanas), donde se desarrollan las funcionalidades del videojuego, su duración debe ser constante en cada ciclo. Sin embargo, este requisito no es estricto debido a que a veces el tiempo de cada iteración es diferente; una Iteración consta de las siguientes actividades:
    - **Elaborar:** Se desarrollan las funcionalidades o requisitos del videojuego que se encuentran en la reserva de iteración.
    - **Integrar:** Las funcionalidades desarrolladas son continuamente integradas en la nueva versión del videojuego.
    - **Revisar:** Se examinan las funcionalidades desarrolladas en la iteración para corregir posibles errores y probar que su funcionalidad sea correcta.
    - **Ajustar:** Se adecua las funcionalidades del videojuego desarrolladas encajándolas correctamente, además se realiza la refactorización del código de las mismas.

Estas actividades no tienen una secuencia. A veces un elemento de la Reserva del Producto se tiene que desarrollar, integrar, y revisar cuando otros sólo deben ser revisado o ajustado. En la Figura #2 se puede observar como es el proceso de las iteraciones.

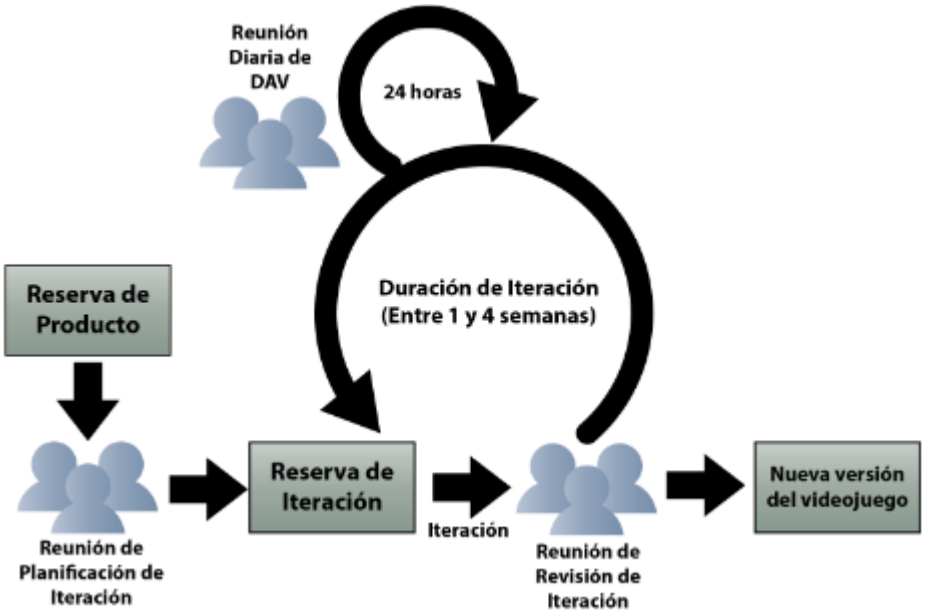


Figura #2: Proceso del DAV

Fuente: Isidro García y Manuel Galán

Al comienzo de cada Iteración se realiza su planificación, donde se describen, priorizan y estiman las funcionalidades que se van a desarrollar, también se efectúan reuniones diarias para mejorar la comunicación del Equipo DAV y al concluir cada Iteración se evalúa su resultado.

***Fase post-juego**

Esta fase comienza cuando el equipo DAV ha desarrollado con éxito todas las Historias de Usuario y el Dueño del Producto ya no tiene más funcionalidades para implementar. Aquí se realiza el cierre del proyecto, donde se prepara la liberación del videojuego, se verifican las versiones a entregar, se genera la documentación final y se realiza el pre-lanzamiento y el lanzamiento.

CAPÍTULO IV

"**Desarrollo de la Propuesta**", en este capítulo se detalla de una manera clara el desarrollo de la propuesta de solución, la metodología de desarrollo de la aplicación, el diseño de la interfaz gráfica de usuario, el diseño de la base de datos y la descripción de los datos, además de la implementación.

6. DESARROLLO DE LA PROPUESTA

Estudio de los principales Frameworks existentes en el mercado tecnológico para el desarrollo de videojuegos para dispositivos móviles.

6.1 Frameworks

Un framework es una estructura de soporte en el cual el desarrollo de software puede ser organizado y encarado con mayor simplicidad. Un buen framework es aquel que resuelve los problemas técnicos más complejos, como, por ejemplo, el manejo de transacciones, permitiendo a los desarrolladores concentrarse en el diseño y desarrollo de la aplicación sin llegar a ser intrusivos para el programador.

Los frameworks suelen incluir: soporte de programas, bibliotecas, lenguaje de scripting, software para desarrollar y unir diferentes componentes de un proyecto de desarrollo de programas.

Esta generación ha sido hogar de una enorme cantidad de videojuegos, los cuales fueron desarrollados sobre motores gráficos muy capaces y altamente superiores a lo que se ha visto en otras épocas. También fue una de las primeras veces en que las grandes empresas empezaron a licenciar sus motores gráficos, como los populares Unreal Engine y el id Tech, dichos motores se han convertido en frameworks completos, ofreciendo un conjunto de herramientas específicas para el desarrollo de juegos, lo que facilita la incursión de pequeñas empresas y desarrolladores independientes.

6.2 Unity3D

Es un framework y motor de juego para el desarrollo de videojuegos multiplataforma creado por Unity Technologies. Unity está disponible como plataforma de desarrollo para Microsoft Windows y OS X, permite crear juegos para Windows, OS X, Linux, Xbox 360, PlayStation 3, PlayStation Vita, Wii, Wii U, iPad, iPhone, Android y Windows Phone.

Unity 3D provee un editor visual muy útil y completo donde mediante unos pocos clicks se puede importar modelos 3D, texturas, sonidos, etc. para después ir trabajando con ellos. Además, incluye la herramienta de desarrollo MonoDevelop con la que podremos crear scripts en JavaScript, C# y un dialecto de Python llamado Boo con los que extender la funcionalidad del editor, utilizando las API que provee la cual se encuentra documentada junto a tutoriales y recursos en su web oficial. Si el desarrollador no domina el diseño en 3D o necesita recursos para su juego, en la propia aplicación se puede acceder a una tienda como si de la App Store se tratase, donde se encontrará multitud de recursos gratuitos y de pago incluso se puede extender la herramienta mediante plugins que se obtendrán en dicha tienda.

6.3 Unreal Engine 4

Unreal Engine es un framework y motor de juego de PC y consolas creados por la compañía Epic Games. Implementado inicialmente en el shooter en primera persona llamado Unreal en el año 1998.

La versión actual está programada en C++ y es compatible tanto con OpenGL como DirectX 11 y 12, siendo compatible con varias plataformas como PC (Microsoft Windows, GNU/Linux), Apple Macintosh (Mac OS X) y la mayoría de consolas (Xbox One y PlayStation 4). Unreal Engine también ofrece varias herramientas de gran ayuda para diseñadores y artistas, facilitando la visualización de entornos o de construcciones.

Unreal Engine 4 ha sido reescrito desde cero en C++. Esto solo tiene un inconveniente:

imposible mudar de UE3 (Unreal Engine 3) a UE4 (Unreal Engine 4) sin exportar a formatos universales y volver a importar los assets e imposible mudar la programación gráfica o tradicional. UE4 directamente fusila el pseudo lenguaje que se inventó Epic Games para UE3 llamado unrealscript (en parte pensado para que el público no pudiera acceder al código fuente del engine) y abre directamente

todo el código fuente de Unreal Engine 4 al público y, por si fuera poco, lo pone de acceso público en Github para que cualquiera pueda revisarlo
y aportar mejoras.

Por otra parte, uno de los mejores elementos de Unreal Engine 3, Kismet, una especie de entorno para programar sin tener que usar código, evoluciona para convertirse en un entorno de programación gráfica con el que se pueden crear hasta clases de cualquier tipo. Ahora sí se puede decir; si quieres puedes construir juegos desde cero sin tener que escribir una sola línea de código. Tan solo uniendo “flechitas”. Eso sí, no hay que pensar que la programación de toda la vida queda relegada a un segundo plano; el código C++ aun es necesario para realizar tareas complejas de programación.

6.4 Godot Engine

Godot es un framework y motor multiplataforma de juego 2D y 3D con licencia MIT de código abierto desarrollado por la comunidad Godot Engine y se utiliza internamente por varias empresas en América Latina antes de ser evaluado y publicado el código del motor. El framework de desarrollo se ejecuta en Windows, OS X y Linux (32 y 64 bits).

Godot soporta múltiples plataformas. Dentro de un proyecto, los desarrolladores tienen control para desplegar en móviles, web, PC, y consolas. Godot también deja especificar la compresión de textura y encuadres de resolución para cada plataforma. Actualmente las plataformas soportadas son Windows, OS X, Linux, Android, iOS, BlackBerry 10, HTML5, PlayStation 3, PlayStation Vita y Nintendo 3DS.

Los videojuegos en Godot son codificados en el lenguaje de programación C++, o en el lenguaje GDScript. GDScript, es un lenguaje de programación de alto nivel, muy similar a Python que fue creado especialmente para Godot, por lo que añade funcionalidades y optimización. Tiene un editor con auto indentación, resaltado de

sintaxis, autocompletado de código y depurador que soporta breakpoints y ejecución paso a paso.

- **Comparación de diferentes motores de juegos tanto de tipo comercial como Open Source para el desarrollo de aplicaciones sobre plataformas móviles.**

Oportunamente se ha concluido que los frameworks para el desarrollo de videojuegos incluyen su propio motor de juego, por lo tanto, desde ahora en adelante al hablar de un framework se estará hablando del framework y del motor como un solo conjunto de herramientas.

Para la evaluación de los frameworks se utilizó la escala de evaluación por puntos donde 0 = no cumple, 1 = cumple parcialmente y 2 cumple totalmente. A continuación, se describen los criterios a evaluar.

- **Licenciamiento gratuito.** - Esta característica se refiere a licencias gratuitas otorgadas a desarrolladores freelance que no obtienen ganancias por la publicación de su producto.
- **Varias plataformas de desarrollo.** - Esta característica se refiere a los sistemas operativos en los cuales se puede instalar el entorno de desarrollo.
- **Requisitos mínimos para desarrolladores.** - Característica que indica requisitos de hardware accesibles para los desarrolladores.
- **Lenguaje de programación de alto nivel.** - Indica los lenguajes de programación de alto nivel aceptados por el framework.
- **Orientación 2D y 3D.-** Esta característica se refiere a la capacidad del motor para desarrollo de videojuegos en 2 y 3 dimensiones.
- **Publicación multiplataforma.** - Característica que indica la posibilidad de publicar el juego en más de una plataforma.

- **Herramientas extras para desarrollo móvil.** - Indica si el framework posee herramientas específicas para el desarrollo de videojuegos móviles.
- **Integración con otras herramientas.** - Se refiere a la posibilidad de integrarse con otras herramientas como software de diseño 3D.
- **Suficiente documentación.** - Esta característica indica la cantidad de documentación disponible en la propia página junto con el contenido de ayuda creado por la comunidad.

6.5 Resultado de la evaluación

Los resultados de la evaluación de los frameworks se muestran a continuación en el siguiente gráfico:

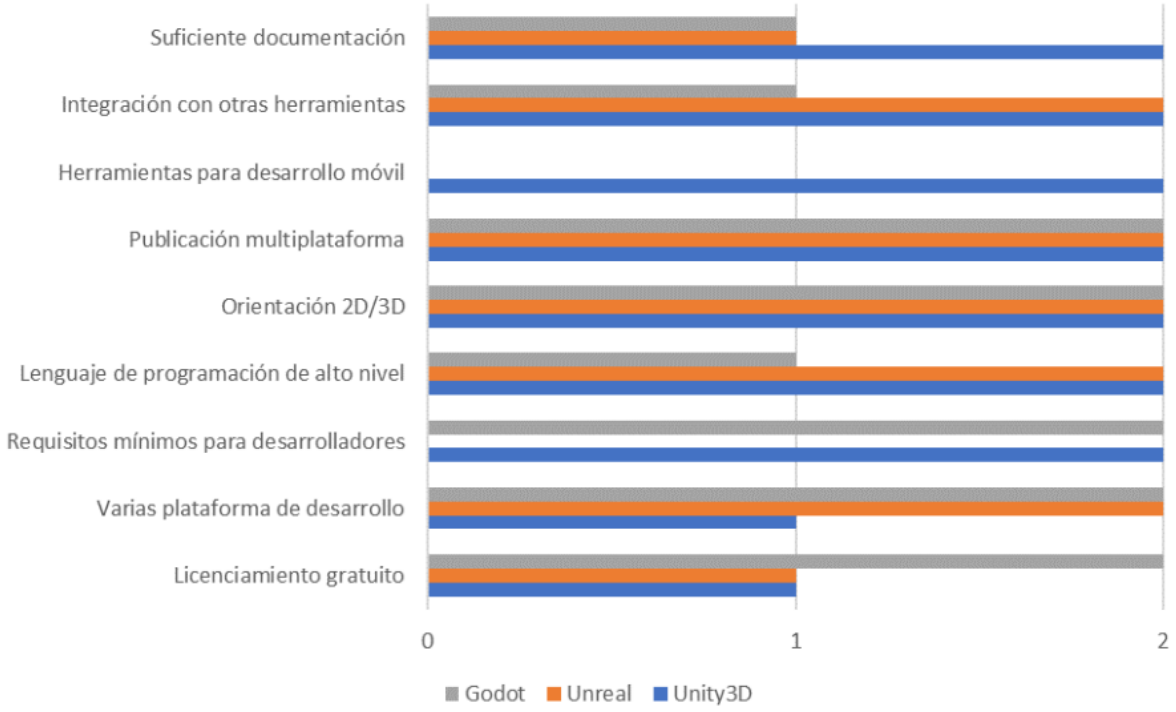


Figura #3: Resultados evaluación frameworks

Fuente: Isidro García y Manuel Galán

A continuación de detalla un breve análisis de los resultados del gráfico anterior.

- **Licenciamiento gratuito.** - Unity3D y Unreal obtienen 1 punto ya que su licencia gratuita tiene condiciones de regalías, por lo que Godot Engine gana en este apartado con 2 puntos al poseer una licencia abierta.
- **Varias plataformas de desarrollo.** - En este criterio gana Unreal y Gotot con 2 puntos cada uno ya que poseen soporte para Windows, Linux y Mac, Unity se queda con un punto ya que no posee soporte para Linux.
- **Requisitos mínimos para desarrolladores.** - Unity3D y Godot empatan en este criterio ya que sus requisitos de hardware son bastante accesibles en comparación de los requisitos necesarios para ejecutar el editor de Unreal, por lo que este obtiene 0 puntos.
- **Lenguaje de programación de alto nivel.** - Los tres productos poseen lenguajes de alto nivel, por lo que obtienen 2 puntos.
- **Orientación 2D y 3D.-** Los tres productos están orientados a los gráficos 2D y 3D, por lo que obtienen 2 puntos.
- **Publicación multiplataforma.** - Los tres productos tienen la posibilidad de publicar en múltiples plataformas, por lo que obtienen 2 puntos.
- **Herramientas extras para desarrollo móvil**. - Solo Unity3D posee herramientas específicas para el desarrollo móvil por lo que obtiene 2 puntos, mientras Unreal y Godot 0 puntos.
- **Integración con otras herramientas.** - Unity3D y Unreal obtienen 2 puntos por su integración con varias herramientas de diseño, mientras que Godot obtiene 1 debido a que solo se integra con una herramienta de diseño.
- **Suficiente documentación.** - Unity3D y Unreal obtienen 2 puntos ya que poseen suficiente documentación tanto oficiales como por su comunidad, mientras que Godot obtiene 1 por su insuficiente documentación.

En base a los resultados mostrados, Unity3D obtiene 16 puntos, Unreal 12 puntos y Godot 13 por lo tanto para este proyecto se ha seleccionado Unity3D como framework y motor de juego apropiado para el desarrollo del videojuego.

6.6 Descripción de los componentes de Unity3D

Editor Unity

La interfaz del editor de Unity después de su instalación y ejecución se puede observar en la Figura 4, cada una de las zonas del editor permiten realizar una tarea específica, la zona central será el lugar en el cual se muestren todos los elementos del juego en ejecución y los botones para la ejecución del juego se encuentran en la parte superior.

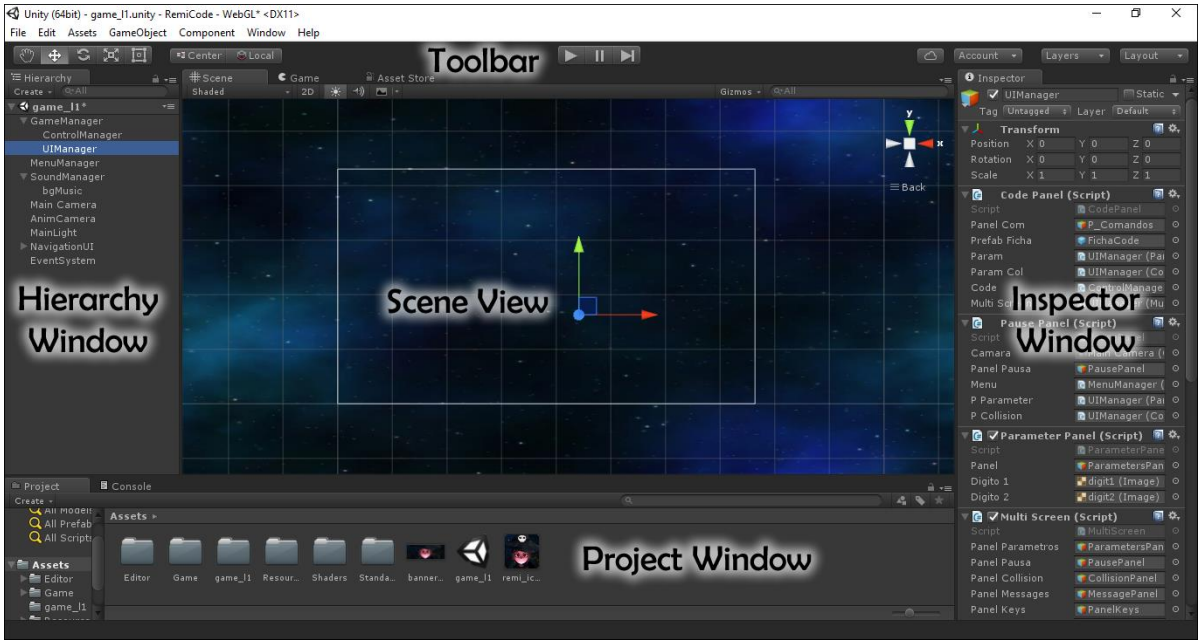


Figura #4: Partes de la interfaz del editor de Unity3D

Fuente: Isidro García y Manuel Galán

- ✓ **Project Windows.** - Muestra los assets de librería que están disponibles para ser usados.

- ✓ **Scene View.** - Permite una navegación visual y editar la escena. La scene view puede mostrar una perspectiva 2D o 3D dependiendo en el tipo de proyecto en el que esté trabajando.

- ✓ **Hierarchy Windows.** - Es una representación de texto jerárquico de cada objeto en la escena; cada elemento en la escena tiene una entrada en la jerarquía, por lo que las dos ventanas están inherentemente vinculadas. La jerarquía revela la estructura de cómo los objetos están agrupados el uno al otro.

- ✓ **Inspector Windows.** - Permite visualizar y editar todas las propiedades del objeto actualmente seleccionado, cada objeto tiene diferentes propiedades por lo tanto el contenido de la ventana va a variar.

- ✓ **Toolbar.** - Proporciona un acceso a las características más esenciales para trabajar; en la izquierda contiene las herramientas básicas para manipular la scene view y los objetos dentro de esta; en el centro están los controles de reproducción, pausa, y pasos; los botones a la derecha dan acceso a servicios de Unity Cloud y una cuenta de Unity, seguido por un menú de visibilidad de capas, y finalmente el menú del layout del editor (que proporciona algunos diseños alternativos para la ventana del editor, y le permite a usted guardar sus propios layouts personalizados).

6.7 MonoDevelop

MonoDevelop es un entorno de desarrollo integrado o entorno de desarrollo interactivo, en inglés Integrated Development Environment (IDE) proporcionado con Unity. Un IDE combina la operación familiar de un editor de texto con características adicionales para depurar y gestionar otras tareas del proyecto.

Este IDE está diseñado primordialmente para C#, Javascript y Boo

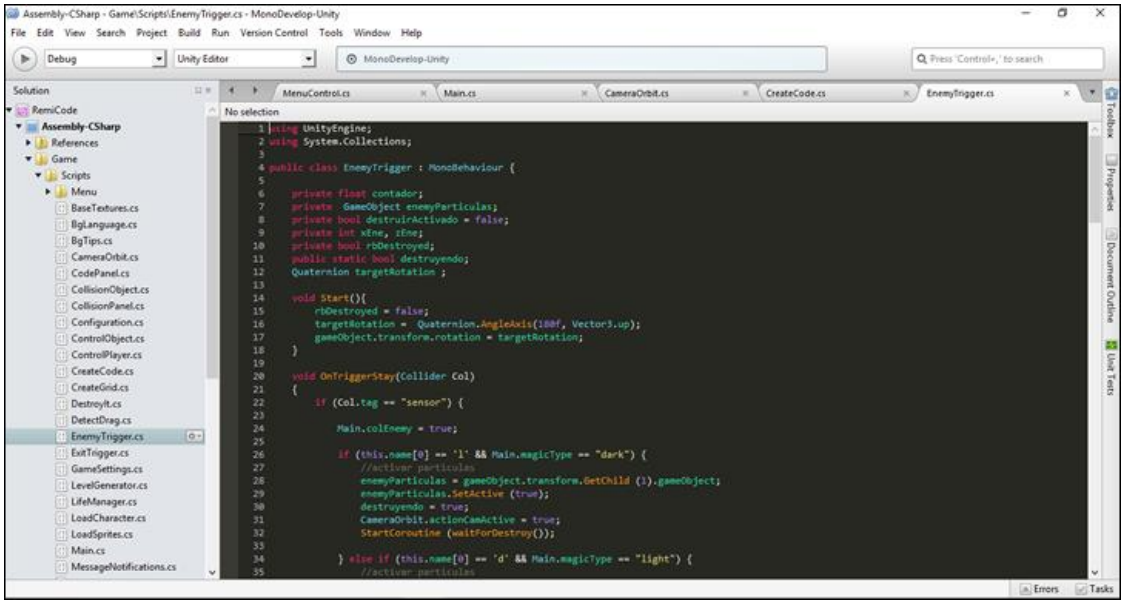


Figura #5: MonoDevelop

Fuente: Isidro García y Manuel Galán

6.8 Arquitectura del motor

La estructura del motor Unity se divide en las siguientes capas como se muestra en la Figura 6:

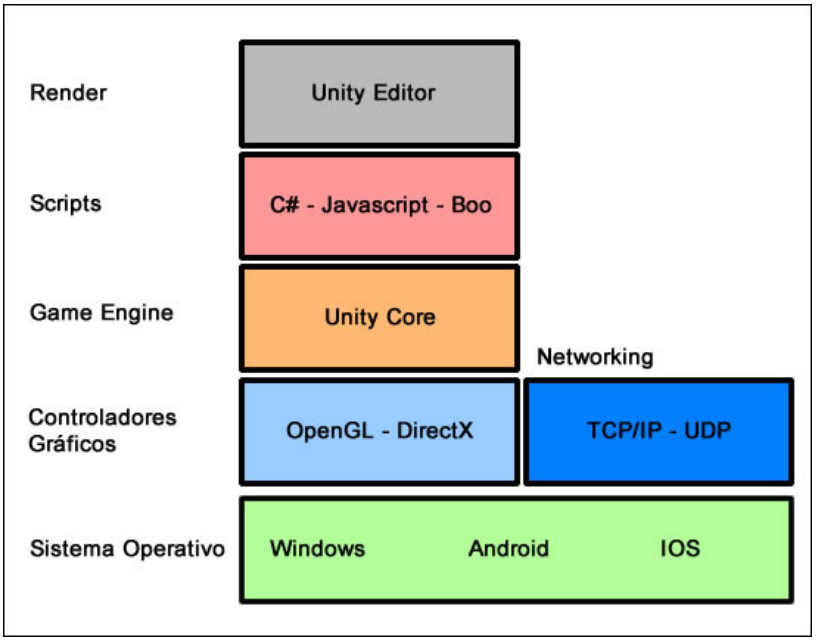


Figura #6: Arquitectura de Unity3D

Fuente: Isidro García y Manuel Galán

- ✓ **Sistema Operativo-** Es la capa en la cual se ejecutará el videojuego, ya sean plataformas móviles como Android e IOS o plataformas de escritorio como Windows.
- ✓ **Controladores Gráficos**-Según la plataforma en la que se ejecute el videojuego, requerirá un controlador gráfico como OpenGL para Android o DirectX para Windows.
- ✓ **Networking**-Esta capa es usada en juegos con características de multijugador utilizando el protocolo TCP/IP (Protocolo de Control de Transmisión/Protocolo de Internet) y UDP (Protocolo de datagrama de usuario).
- ✓ **Game Engine**-Es el núcleo de toda la arquitectura, el cual contiene los diferentes componentes que se utilizarán para la creación del juego, componentes como físicas, colisiones, animaciones, materiales, texturas, etc.
- ✓ **Scripts**-La programación y la lógica del juego se realiza aquí utilizando lenguajes como C#, javascript y Boo equivalente a Python.
- ✓ **Render**-La última capa en la cual se muestra el mundo/nivel de forma jugable desde el Editor de Unity

6.9 Selección de una metodología ágil aplicada al campo del desarrollo de videojuegos móviles que facilite el desarrollo e implementación del proyecto planteado.

Desarrollar un proyecto de software requiere de una metodología apropiada para una mejor organización y planificación de las diferentes tareas que se llevarán a cabo por el equipo de desarrollo, lo que permitirá tener un control de lo que se está haciendo y lo que aún falta hacer, evitando el fracaso del proyecto al tener estimaciones de tiempo totalmente incorrectas o recursos mal utilizados.

Un videojuego es también un proyecto de software en el cual interviene un equipo multidisciplinario con conocimientos específicos en áreas como: programación, diseño gráfico y multimedia, efectos de sonidos, etc., por lo tanto, el uso de una buena metodología determinará el éxito o fracaso del proyecto.

Como se ha mencionado anteriormente, las pequeñas empresas desarrolladoras de videojuegos con equipos realmente pequeños no cuentan con una metodología formalizada y recurren principalmente al uso de metodologías ágiles como SCRUM y XP adaptándolas a sus necesidades.

Estas metodologías se han transformado en metodologías ágiles específicamente para el desarrollo de videojuegos, y es aquí donde nacen nuevas metodologías como SUM, basándose en la estructura de SCRUM; aparece también XGD (Extreme Game Development) la cual es una adaptación de eXtreme Programming (XP) para videojuegos; adicionalmente a estas dos metodologías aparece una tercera y muy interesante que es la combinación entre las características de SCRUM y XGD, llamada metodología DAV (Desarrollo Ágil de Videojuegos).

***Metodología SUM**

La metodología SUM para videojuegos tiene como objetivo desarrollar videojuegos de calidad en tiempo y costo, así como la mejora continua del proceso para incrementar su eficacia y eficiencia. Pretende obtener resultados predecibles, administrar los recursos y riesgos del proyecto, y lograr una alta productividad del equipo de desarrollo. SUM fue concebida para que se adapte a equipos multidisciplinarios pequeños (de tres a siete integrantes que trabajan en un mismo lugar físico o estén distribuidos), y para proyectos cortos (menores a un año de duración) con alto grado de participación del cliente.

Roles

La metodología define cuatro roles: equipo de desarrollo, productor interno, cliente y verificador beta. El productor interno y el cliente se relacionan en forma directa con los roles de SCRUM Master y Product Owner de SCRUM respectivamente.

El equipo de desarrollo tiene las características del Scrum team, pero a diferencia de SCRUM se definen subroles dentro del equipo, estos son equivalentes a los

usados por la industria local tales como: programador, artista gráfico, artista sonoro y diseñador de juego. Es necesario esta definición ya que se requiere una alta especialización para satisfacer las distintas disciplinas que involucra del desarrollo de videojuegos, aspecto no contemplado en Scrum.

El rol de verificador beta no está presente en Scrum pero si se detecta su existencia en la industria del videojuego. Su responsabilidad es realizar la verificación funcional del videojuego y comunicar su resultado.

Los objetivos de cada uno de los roles son los siguientes:

1. **Cliente.** - Define y valida el concepto del juego, aprueba los planes de proyecto y determina los hitos, prioriza las características y tareas que dan más valor al videojuego en cada momento, evalúa el cumplimiento de las tareas y el producto obtenido al finalizar cada iteración, y participa de la evaluación del proyecto. Prioriza los errores a corregir buscando la mejor calidad posible de acuerdo a sus intereses y valida las versiones del producto.

2. **Productor Interno**- Es responsable de la planificación y la ejecución del proyecto, participa en la definición de los objetivos y hace seguimiento del proyecto ayudando a resolver los impedimentos que ocurren en el proyecto. Lleva a cabo las acciones para la mejora continúa coordinando la comunicación con el cliente y mantiene al equipo enfocado en los objetivos del proyecto.

3. **Equipo de Desarrollo**. - Representa a todos los miembros encargados de construir el videojuego.
    - **Diseñador de juego**. - Se encarga de diseñar el gameplay, la historia, ambientación, personajes, niveles y todos los elementos que hacen a la experiencia del jugador. Todos estos factores determinarán que tan divertido será el juego. Para asegurar la diversión debe mantener balanceada la dificultad del juego y el aprendizaje del jugador.

- **Programador.** - El programador tiene como principal responsabilidad implementar el software que compone al juego. Además, deberá realizar el diseño de software necesario para poder realizar el desarrollo y posteriormente verificarlo. Por lo tanto, el desarrollador de videojuegos debe tener conocimientos de diseño de software, implementación y verificación.
- **Artista sonoro**. - Los artistas sonoros deben tener buen oído para mezclar los sonidos, los efectos de sonido deben ser diseñados de forma que se correspondan con la interacción del jugador. El sonido da vida a la escena y complementa la experiencia del jugador.
- **Artista gráfico**. - El arte y la animación son gran parte del trabajo requerido para el desarrollo del videojuego. Las habilidades necesarias para un artista varían según los requerimientos del juego en particular. De cualquier forma, requieren conocimientos sobre las últimas herramientas gráficas, creatividad y talento.

4. Verificador Beta. - Son los designados para realizar la verificación funcional del videojuego. Participan, fundamentalmente, en la etapa beta del proyecto, ya que es cuando se obtiene la primera versión completa del juego. Dependiendo del proyecto es posible que participen verificadores beta un poco antes de dicha etapa.

Ciclo de vida

El ciclo de vida se divide en fases iterativas e incrementales que se ejecutan en forma secuencial con excepción de la fase de gestión de riesgos que se realiza durante todo el proyecto. Las cinco fases secuenciales son: concepto, planificación, elaboración, beta y cierre, como se aprecia en la Figura. Las fases de concepto, planificación y cierre se realizan en una única iteración, mientras que elaboración y beta constan de múltiples iteraciones.

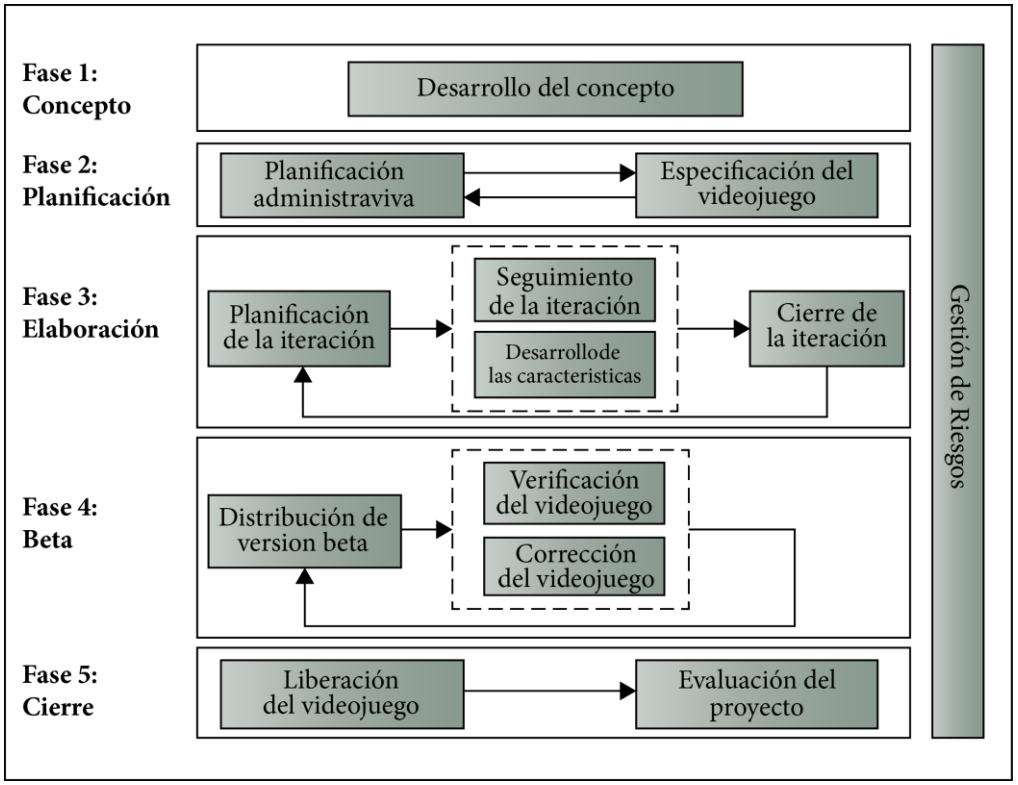


Figura #7: Ciclo de vida de la metodología SUM

Fuente: Isidro García y Manuel Galán

1. Fase del concepto

Tiene como objetivo principal definir el concepto del videojuego lo que implica definir aspectos de negocio (público objetivo, modelo de negocio), de elementos de juego (principales características, gameplay, personajes e historia entre otros) y técnicos (lenguajes y herramientas para el desarrollo). El concepto del videojuego se construye a partir de ideas y propuestas de cada rol involucrado sobre los aspectos a definir. Las propuestas se refinan a través de reuniones y se analiza su factibilidad con pruebas de concepto.

Esta fase finaliza cuando se tiene el concepto validado entre todas las partes involucradas.

2. Fase de la planificación

La fase tiene como objetivo principal planificar las restantes fases del proyecto. Para ello es necesario definir el cronograma del proyecto junto con sus principales hitos, conformar el equipo para la fase de elaboración de acuerdo a las necesidades técnicas del proyecto, determinar y tercerizar las tareas que el equipo no pueda cumplir, definir el presupuesto y especificar las características. Esto último consiste en describir, estimar y priorizar cada una de las características funcionales y no funcionales que definen el videojuego.

3. Fase de elaboración

El objetivo de esta fase es implementar el videojuego, para ello se trabaja en forma iterativa e incremental para lograr una versión ejecutable del videojuego al finalizar cada iteración. Estas se dividen en tres etapas, en la primera se planifican los objetivos a cumplir, las métricas a utilizar en el seguimiento, las características a implementar y las tareas necesarias para ello. En la segunda se desarrollan las características planificadas a través de la ejecución de las tareas que la componen.

4. Fase beta

La fase tiene como objetivos evaluar y ajustar distintos aspectos del videojuego como por ejemplo gameplay, diversión, curva de aprendizaje y curva de dificultad, además de eliminar la mayor cantidad de errores detectados. Se trabaja en forma iterativa liberando distintas versiones del videojuego para verificar. Para ello primero se distribuye la versión beta del videojuego a verificar y se determinan los aspectos a evaluar. Mientras esta se verifica, se envían reportes con los errores o evaluaciones realizadas.

5. Fase de cierre

Esta fase tiene como objetivo entregar la versión final del videojuego al cliente según las formas establecidas y evaluar el desarrollo del proyecto. Para la

evaluación se estudian los problemas ocurridos, los éxitos conseguidos, las soluciones halladas, el cumplimiento de objetivos y la certeza de las estimaciones. Con las conclusiones extraídas se registran las lecciones aprendidas y se plantean mejoras a la metodología.

6. Fase de gestión de riesgos

Esta fase se realiza durante todo el proyecto con el objetivo de minimizar la ocurrencia y el impacto de problemas. Esto se debe a que distintos riesgos pueden ocurrir en cualquiera de las fases, por lo cual siempre debe existir un seguimiento de los mismos. Para cada uno de los riesgos que se identifican se debe establecer la probabilidad y el impacto de ocurrencia, mecanismos de monitoreo, estrategia de mitigación y plan de contingencia.

Metodología DAV

DAV (Desarrollo Ágil de Videojuegos), es una metodología que nace con la unión de las características de SCRUM y XGD, con la finalidad de proporcionar una estructura que permita la creación de videojuegos de forma técnica y sencilla. Esta metodología se compone de los siguientes elementos: valores, prácticas, roles, reuniones, fases, artefactos y herramientas.

Prácticas

Esta metodología utiliza prácticas tanto de SCRUM como de XGD para integrar en la nueva metodología DAV, a continuación, se describe cada una de ellas:

- Diseño incremental: Las tareas de un videojuego deben ser de la forma más sencilla posible, que funcionen correctamente.

- Historias de usuario: Se trata de una breve descripción de las funcionalidades del videojuego, por lo general descrito por el Dueño del Producto.

- Ciclos semanales: El proyecto está organizado y planificado para ser ejecutado en ciclos de corta duración.

- Entregas pequeñas: Producir rápidamente versiones del videojuego estables, aunque no cuenten con toda la funcionalidad del videojuego. Esta versión ya constituirá un resultado de valor para el negocio.

- Pruebas: El videojuego será probado en varios momentos por diferentes personas, dando como resultado una información valiosa que podrá ser tomada como base para decisiones futuras.

*** Roles**

Los roles de DAV se basan únicamente en SCRUM, añadiendo al equipo DAV el rol de tester, siendo estos necesarios y suficientes para ser aplicados en equipos pequeños, clasificándose en dos tipos de roles:

Dueño del Producto- El Dueño del Producto es la única persona autorizada para decidir sobre cuáles funcionalidades y características funcionales tendrá el videojuego. Es quien representa al cliente y todas aquellas partes interesadas en el videojuego, además es quien maneja y prioriza las funcionalidades a desarrollar y las coloca en la Reserva de Producto.

DAV Máster- El DAV Máster no es un líder típico, sino un auténtico servidor neutral, que será el encargado de enseñar la metodología DAV a cada integrante implicado en el proyecto, preocupándose de poner la metodología en práctica de modo que se encuentre dentro de la cultura de la organización y así entregue las ventajas previstas, asegurándose de que cada uno siga las reglas y prácticas de DAV:

Equipo DAV- Es un equipo multidisciplinario y auto-organizado, su función principal es construir el videojuego que el Dueño del Producto específico. Los miembros del equipo pueden ser de 3 a 9 personas, entre los cuales tenemos:

• **Diseñador**- Realiza el diseño conceptual, define las reglas y la mecánica, escribe el guion y diseña los niveles.

• **Programador**- Implementa la funcionalidad pedida en el diseño, programa la lógica del juego, gráficos (efectos), inteligencia artificial y audio y conoce de herramientas/motores (infraestructura).

• **Artista-** Construye los objetos, personajes, niveles y anima los personajes.

• **Sonidista**- Crea los sonidos, estilos, efectos y ambientes sonoros para el videojuego.

• **Tester**- Verifican que el contenido, funcionalidad del videojuego son correctos, completos y evalúan la jugabilidad.

Prueba de Aceptación	
Identificador: PA001	**Historia de Usuario (Nro. y Nombre):** HU001-Visualizar las Instrucciones del videojuego
Nombre: Visualizar las instrucciones de usuario.	
Descripción: Visualizar en una ventana las instrucciones del videojuego, además mediante un botón de color amarillo debe permitir regresar al menú principal.	
Condiciones de ejecución: Debe ejecutarse en el dispositivo móvil, además debe estar terminado el menú inicio.	
Entrada/Pasos de ejecución: - Escoger la opción "Instrucciones" en el Menú Principal. - En la pantalla que aparece las instrucciones del videojuego, para regresar al Menú Principal dar clic en el botón Regresar.	
Resultado esperado: Mostrar las instrucciones del videojuego en una ventana, además tener un botón que permita salir de la ventana y regresar al menú principal.	
Evaluación de la prueba: Correcto	

Tabla #1: Prueba de aceptación de DAV

Fuente: Isidro García y Manuel Galán

Para cada ítem, es necesario especificar:

- **Identificador:** Es un identificador único de la Prueba de Aceptación para futuras referencias.
- **Historia de usuario (Nro. Y Nombre):** El identificador y nombre de la Historia de Usuario a la cual pertenece la Prueba de Aceptación.
- **Nombre:** La Prueba de Aceptación debe tener un nombre entendible para cualquier persona, facilitando la comprensión tanto del propósito de la prueba como del campo de aplicación.
- **Descripción:** Contiene una breve descripción del propósito de la prueba y la funcionalidad que examina.
- **Condiciones de ejecución:** Contiene información acerca de hardware o software necesario para poder ejecutar la prueba.
- **Entrada / pasos de ejecución:** Pasos a realizar para completar la prueba.
- **Resultado esperado:** Contiene una descripción de lo que el analista debería ver tras haber completado todos los pasos de la prueba.
- **Evaluación de la prueba:** Indica el resultado cualitativo de la ejecución de la prueba, a menudo con un Correcto/Fallido.

Reserva del producto

La Reserva del Producto es un listado de alto nivel, dinámico y públicamente visible para todos los involucrados en el proyecto. En ella, el Dueño de Producto, mantiene una lista actualizada de requerimientos funcionales para el videojuego.

Esta lista, representa “qué es lo que se pretende hacer” pero sin mencionar “cómo hacerlo”, debido a que esta última, será tarea del Equipo DAV. La Reserva del Producto es creada y modificada únicamente por el Dueño de Producto. Durante la reunión de planificación, el Equipo DAV obtendrá los ítems del videojuego, que deberá desarrollar durante la Iteración. En la siguiente tabla se muestra el formato de la Reserva del Producto.

PRI	Nro. HU	Ítem	Estimación	Estado	PA
1	HU01	Como Jugador quiero visualizar las instrucciones del juego, para saber cómo jugar.	3	Pendiente	PA01
2	HU02	Como jugador quiero visualizar el puntaje de los bloques que voy rompiendo, de modo que pueda conocer cuántos puntos voy acumulando.	5	Pendiente	PA01

Tabla #2: Reserva producto de DAV

Fuente: Isidro García y Manuel Galán

Para cada ítem, es necesario especificar:

El grado de prioridad (PRI).- Los ítems de la Reserva del Producto, deben guardar un orden de prioridad, en base a la pérdida o costo que demande posponer la implementación.

Esfuerzo que demanda.- DAV, propone la estimación de esfuerzo y complejidad que demanda el desarrollo de las funcionalidades, solo para aquellas cuyo orden sea prioritario. Estas estimaciones, no se efectúan sobre ítems poco prioritarios ni tampoco sobre aquellos donde exista un alto grado de incertidumbre. De esta manera, se evita la pérdida de tiempo en estimaciones irrelevantes, postergándolas para el momento en el cual realmente sea necesario comenzar a desarrollarlas.

Prueba de aceptación (PA).- Es necesario especificar para cada ítem de la Reserva del Producto, la o las Pruebas de Aceptación que debe superar, para considerar cumplido el requisito.

Tarjeta de tarea

Las Tarjetas de Tarea se usan para describir las tareas que se realizarán en el proyecto y representar las responsabilidades asignadas a cada miembro del Equipo DAV. Estas tareas pueden ser de desarrollo, corrección o mejora, contienen número y nombre de la tarea, Historia de Usuario a desarrollar, fecha de inicio y fin de la tarea, se nombra al miembro del equipo responsable y presenta una descripción de la tarea como se muestra en la siguiente tabla.

Tarjeta de tarea	
Número de Tarea:	**Historia de Usuario (Nro. y Nombre):**
Nombre Tarea:	
Tipo de Tarea:	**Puntos Estimados:**
Fecha de inicio:	**Fecha fin:**
Miembro responsable:	
Descripción:	

Tabla #3: Tarjeta de tarea de DAV

Fuente: Isidro García y Manuel Galán

Evaluación de las metodologías ágiles SUM y DAV Para la evaluación de las metodología ágiles para desarrollo de videojuegos se utilizó la escala de evaluación por puntos donde 0 = no cumple, 1 = cumple parcialmente y 2 cumple totalmente. A continuación se describen los criterios a evaluar.

Flexibilidad.- Esta característica se refiere a la flexibilidad para cambios durante el desarrollo del proyecto.

Equipos multidiciplinarios.- Esta característica indica la posibilidad de trabajar en equipos con especialistas de diferentes áreas.

Documentación.- La metodología posee una documentación clara y con ejemplos prácticos.

Corto tiempo.- La metodología permite tener versiones entregables del proyecto en corto tiempo.

Comunicación.- La metodología permite tener una comunicación transparente con todos los miembros del equipo, retroalimentando constantemente el estado del proyecto.

Metodología SUM

Características	**Metodología SUM**	
Flexibilidad	Al ser una metodología basada unicamente en SCRUM las tareas terminadas y revisadas no se vuelven a tocar.	0
Equipos multidisciplinarios	Esta metodología posee un rol de equipo multidisciplinario adaptándose al desarrollo de videojuegos.	2
Documentación	Aunque posee una amplia documentación en su página web, no existen ejemplos prácticos para un mejor entendimiento.	1
Corto tiempo	Permite tener entregables del proyecto en poco tiempo para la revisión y aprobación del cliente	2
Comunicación	Posee una comunicación transparente gracias a las reuniones diarias y la retroalimentación del estado del proyecto.	2
Total		7

Tabla #4: Evaluación de la metodología SUM

Fuente: Isidro García y Manuel Galán

Metodología DAV

Características	Metodología DAV	
Flexibilidad	Al basarse en XGD toma las características de esta metodología, permitiendo que las tareas siempre estén susceptibles a cambios aunque ya estén finalizadas y aprobadas.	2
Equipos multidisciplinarios	Esta metodología posee un rol de equipo multidisciplinario adaptándose al desarrollo de videojuegos.	2
Documentación	Posee una amplia documentación y ejemplos prácticos para un mejor entendimiento del proceso de la metodología.	2
Corto tiempo	Permite tener una versión del proyecto en corto tiempo para la revisión del cliente	2
Comunicación	Existe comunicación transparente dentro de todos los miembros del equipo gracias a las reuniones diarias y la retroalimentación del estado del proyecto.	2
Total		10

Tabla #4: Evaluación de la metodología SUM

Fuente: Isidro García y Manuel Galán

Resultado de la evaluación de metodologías

A continuación de detalla un breve análisis de los resultados obtenidos durante la evaluación de la metodología SUM y DAV:

Flexibilidad. - DAV obtiene 2 puntos debido a que permite realizar cambios, aunque la tarea esta finalizada y aprobada, algo muy útil por los cambios constantes en el desarrollo de videojuegos, por otra parte, SUM obtiene 0 puntos ya que no tiene este grado e flexibilidad en tareas finalizadas.

Equipos multidiciplinarios. - Tanto SUM como DAV obtienen 2 puntos ya que permiten la integración de equipos multidiciplinarios.

Documentación. - SUM obtiene 1 punto por la falta de ejemplos prácticos en su metodología, mientras que DAV obtiene 2 puntos por la cantidad de ejemplos documentados que facilitan el uso de esta metodología, tomando en cuenta que ambas metodologías no son usadas comercialmente la documentación es muy importante para aplicar estas herramientas apropiadamente.

Corto tiempo. - Tanto SUM como DAV obtienen 2 puntos ya que permiten realizar entregas del videojuego en corto tiempo.

Comunicación. - Tanto SUM como DAV obtienen 2 puntos ya que permiten una comunicación transparente por todos los miembros del equipo.

En base a los resultados mostrados, SUM obtiene 7 puntos, y DAV 10 puntos, destacándose esta última en su flexibilidad a la hora de realizar cambios a funciones ya terminadas y su documentación ejemplificada algo muy importante en metodologías poco conocidas, por lo tanto, para este proyecto se ha seleccionado DAV como metodología de desarrollo ágil para el desarrollo del videojuego.

Implementación del videojuego educativo en 3d para dispositivos móviles android, para el aprendizaje de la lógica de programación para los estudiantes de la carrera de ingeniería en computación de primero y segundo año.

Se ha seleccionado a la metodología DAV como la más apropiada para el desarrollo de la propuesta de videojuego.

A continuación, se explicará todo el proceso de desarrollo dividido en tres fases principales según como lo indica la metodología.

Fase pre-juego

Descripción del videojuego

El videojuego a desarrollar se llamará “Hello Remi”, el nombre está inspirado en la clásica frase que se imprime en pantalla al aprender un nuevo lenguaje de programación por primera vez “Hello World”, el modo de juego se basa en el concepto de “LightBot”.

El mundo será desarrollado totalmente en 3 dimensiones tanto los personajes como los escenarios, la vista del mundo tendrá dos perspectivas; la primera será una vista aérea que permitirá visualizar todo el mapa del juego; la segunda perspectiva será una vista en tercera persona, permitiendo tener una visualización más cercana de los movimientos y acciones del personaje.

“Remi” será el personaje principal y único personaje jugable, mientras que los personajes NPC (non playable character) serán dos; el primer NPC será un “gato” el cual deberá ser rescatado por el jugador, el segundo NPC será el o los “monstruos” que deberán ser eliminados por el jugador.

El jugador podrá mover a “Remi” únicamente por las casillas del mapa usando acciones predefinidas (mover, girar, volar, aterrizar, usar portal, usar magia).

Las acciones se deberán agrupar dentro de bloques de acciones, existen 3 tipos de bloques:

- ✓ **bloque GO**- Las acciones dentro de este bloque se realizarán una sola vez;
- ✓ **bloque R**- Las acciones dentro de este bloque se realizarán n veces;
- ✓ **bloque IF**- Las acciones de este bloque se realizarán solo si una condición resulta verdadera.

Existirán 3 niveles de dificultad (fácil, medio y difícil), dependiendo de la dificultad el mapa tendrá una mayor cantidad de obstáculos y enemigos, el mapa del juego estará compuesto de columnas que se generarán aleatoriamente al empezar la partida.

Aspectos fundamentales

1. **Género**

Este videojuego entra en la categoría de los puzzles y la estrategia, el jugador deberá utilizar su orientación, razón espacial, razonamiento geométrico y rapidez para determinar el camino correcto a seguir.

2. **Historia**

La historia empieza con Remi y su gato descansando tranquilamente en el bosque, de repente un extraño ruido los asusta, el gato corre hacia el lugar del cual se produjo el ruido, rápidamente aparece una extraña mano monstruosa que se lleva al gato dentro de un oscuro pozo. La misión de la protagonista será entrar al extraño mundo donde se encuentra secuestrado su gato eliminar los monstruos y rescatarlo.

3. **Bocetos**

Los bocetos o diseños preliminares son dibujos realizados a mano de los elementos principales que conformarán el videojuego, hay que tener en cuenta que estos diseños pueden sufrir cambios en la versión final del videojuego por lo tanto deben ser tomados en cuenta como una referencia para el diseñador del juego el cual podrá realizar cambios mínimos si lo requiere necesario.

4. **Aspectos gráficos**

El juego será desarrollado totalmente en 3D constará de 2 cámaras y 4 vistas; la cámara principal mostrará una vista aérea; en tercera persona con rotación manual y una vista 360º con rotación automática la cual se

habilitará con ciertas acciones del jugador, la cámara secundaria mostrará el mapa del juego con una visión de 360º rotando automáticamente, esta cámara se activará únicamente al inicio y al final de la partida.

5. Interfaz de usuario

Todas las acciones se realizan desde la interfaz de usuario (UI: User Interface) que será totalmente táctil, y se dividirá en 5 interfaces como se describen a continuación:

- **UI Logo**. - Esta será la primera interfaz en mostrarse, cuyo único propósito es mostrar el logo del creador del juego.
- **UI Menú principal**. - El menú principal del juego tendrá 4 botones principales: iniciar juego, salir del juego, configuración y créditos.
- **UI StoryBoard**- Esta interfaz se encargará de mostrar un pequeño cómic en forma de storyboard con la introducción de la historia del juego.
- **UI Selección dificultad**- El propósito de esta interfaz será mostrar un menú para seleccionar la dificultad y generar el mundo.
- **UI Juego**- Este interfaz tendrá dos paneles principales, el panel de acciones, el cual contendrá todas las acciones disponibles; el panel de ejecución ubicado a la derecha mostrará los comandos que están siendo ingresados por el jugador, incluye un botón para ejecución y un botón para borrar el ultimo comando ingresado; en la parte superior central se encontrará el botón pausa y en la parte inferior central el botón para cambiar la perspectiva de la cámara.

Características de los elementos del videojuego

En la Tabla 5 se describe de forma resumida las características de los principales elementos del juego.

Elementos	Descripción
Personaje principal (Remi)	Aparece al inicio de la partida, ubicada en la casilla inicial del mapa, se moverá de acuerdo a los comandos que utilice el jugador..
NPC (gato)	El NPC del juego no es un personaje controlable, y su función es la de permitir al jugador terminar la partida
NPC (enemigos)	Existirán dos tipos de enemigos, negros y blancos, los cuales tratarán de impedir que el jugador pueda avanzar libremente por el mata, disminuyendo su vida en un punto si el jugador colisiona con ellos.
PowerUps (calaveras)	Los powerUps son ítems que permiten obtener el poder de magia negra y blanca, algo necesario para derrotar a los enemigos.
Magia (blanca/negra)	Existirá dos tipos de magia, negra y blanca, este poder se obtendrá con 3 PowerUps, permitiendo eliminar enemigos, usando la magia negra, para enemigos blancos y la magia blanca para enemigos negros.
Mundo/Niveles	Cada nivel se generará aleatoriamente.

Tabla #5: Características de los elementos del juego

Fuente: Isidro García y Manuel Galán

7 Objetivo del videojuego

El objetivo del juego es mover al personaje principal del juego “Remi” para obtener 3 powerUps y eliminar a todos los NPC (monstruos) que se encuentran en el escenario, luego de eliminarlos se desbloqueará la ubicación del NPC (gato) que el jugador deberá recuperar. Se deberá terminar el nivel en el menor tiempo posible para obtener una mejor puntuación, además dispondrá de un número de intentos limitados según el nivel de dificultad escogido.

El videojuego busca divertir al jugador mientras mejora su razonamiento y rapidez mental con el uso de diferentes conceptos que se utilizan en la lógica de la programación, pero de forma amigable y divertida.

8. Reglas

El juego inicia con 3 vidas (corazones) 0 powerUps (calaveras) y las acciones de magia deshabilitadas.

- Cada nivel (fácil, medio o difícil) tiene 3 intentos.
- Al dar clic en el botón ejecutar se consumirá un intento.
- Si el jugador colisiona con un monstruo perderá una vida.
- Si el jugador colisiona con un objeto del mapa perderá un intento.
- Si el jugador ingresa una cantidad de pasos que excede al número de casillas perderá un intento.
- Si las vidas del jugador llegan a 0 perderá la partida y deberá empezar de nuevo.
- Si los intentos del jugador llegan a 0 sin haber llegado a la meta, perderá la partida y deberá empezar de nuevo.
- Cuando el jugador colisione con un powerUp sus powerUps se incrementarán en 1.
- Cuando el jugador obtenga 3 powerUps se habilitarán las acciones magia negra/blanca.
- La magia es la única acción que elimina a los monstruos.
- Existen dos tipos de magia y dos tipos de enemigos, blancos y negros.
- Los enemigos blancos se destruyen con magia negra, mientras que los negros se destruyen con magia blanca.
- Existirán enemigos especiales que intercambiarán su color entre negro y blanco.
- Cuando el jugador elimine a todos los enemigos se desbloqueará la ubicación del personaje NPC gato.
- Una vez que el jugador recupere al NPC gato se completará el nivel y se mostrará el puntaje obtenido en base a los intentos utilizados.
- Terminar el nivel con un intento equivale a 3 puntos, dos intentos son 2 puntos y tres intentos son 1 punto.
- Jugar en dificultad fácil multiplicará los puntos por uno, nivel medio por 2 y nivel difícil por 3.

- Los puntos obtenidos permiten desbloquear nuevos trajes para el personaje.

Aspectos técnicos

1. Requerimientos técnicos

Los requisitos técnicos, tanto de hardware como de software son los siguientes:

Hardware	Software
▪ CPU ARMv7 (Cortex) con tecnología NEON o CPU Atom; OpenGL ES 2.0 o posterior. ▪ 1GB Ram	Android OS 2.3.3 o superior

Tabla #6: Requisitos técnicos

Fuente: Isidro García y Manuel Galán

2. Arquitectura del videojuego

La arquitectura del videojuego utiliza MVC (Modelo Vista Controlador) aplicado específicamente a este proyecto y al motor Unity, como se muestra a continuación:

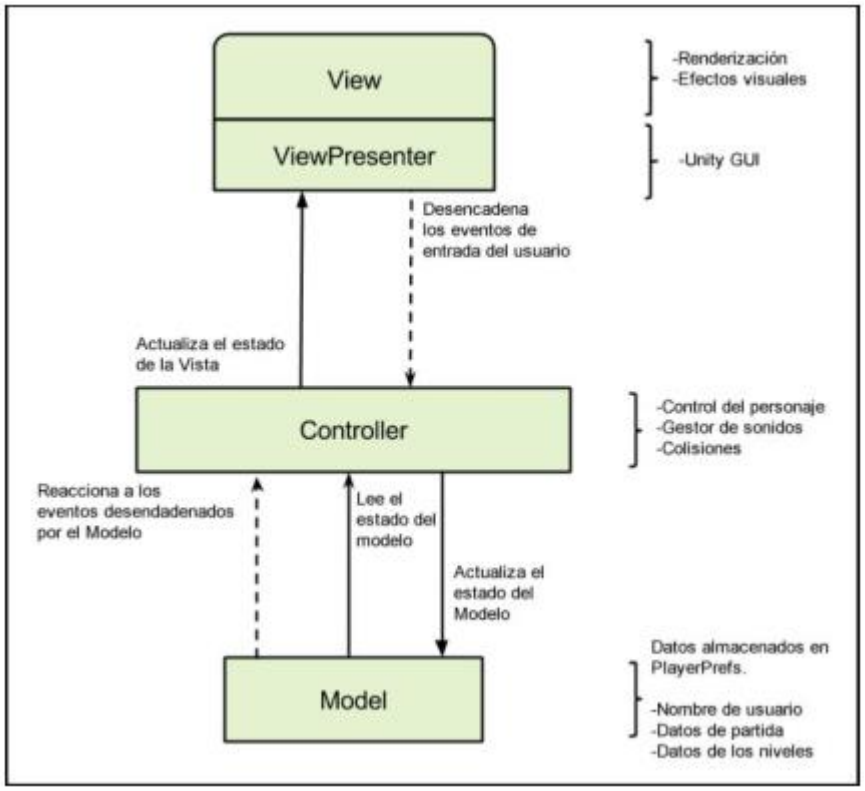


Tabla #7: MVC del videojuego

Fuente: Isidro García y Manuel Galán

Model.- Esta capa se encarga de gestionar los datos que serán usados durante la ejecución del videojuego utilizando la clase PlayerPrefs de Unity, encargada de guardar y mostrar datos.

Controller.- Esta capa se encarga de leer los datos almacenados en el Modelo para actualizar el estado de la Vista, gestionando otros componentes del videojuego como el control del personaje, la gestión de los sonidos y las colisiones.

ViewPresenter.- Esta capa muestra la interfaz de usuario GUI (Interfaz gráfica de usuario), y desencadena los eventos de entrada del usuario que son enviados hacia la capa del Controlador.

View.- La ultima capa se encarga de la renderización de los elementos del videojuego, así como la gestión de efectos visuales.

3. Diseño conceptual

En la Figura 8 se indica el diseño conceptual que contiene las diferentes interfaces de usuario y su forma de navegación.

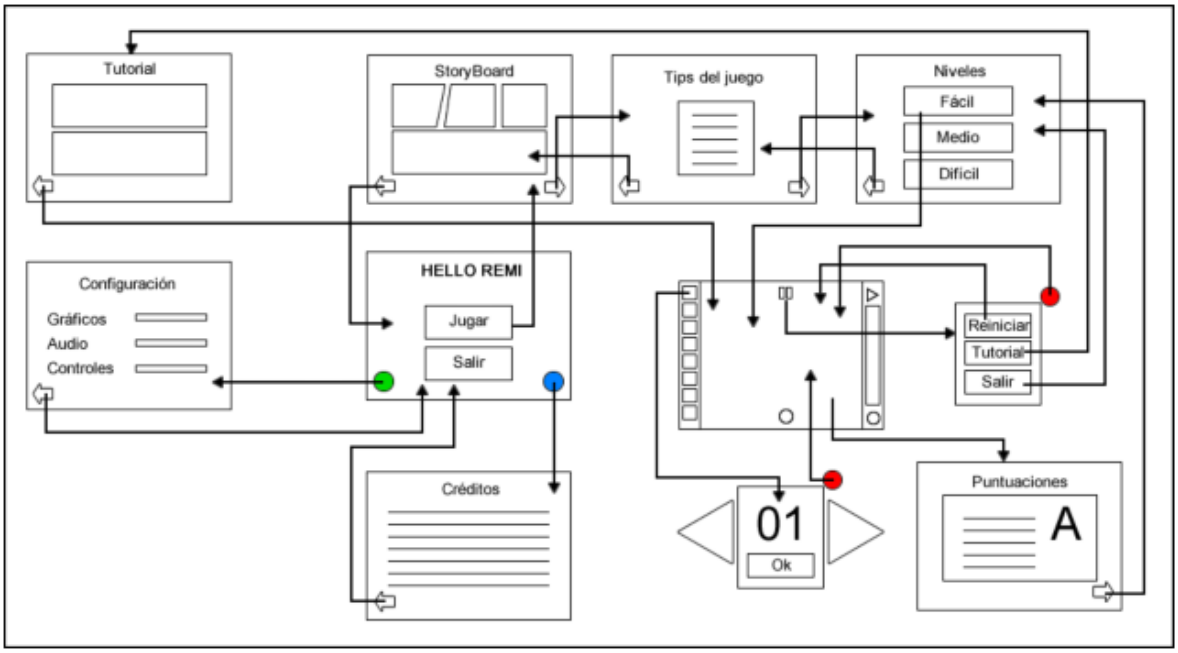


Tabla #8: Diseño conceptual
Fuente: Isidro García y Manuel Galán

Herramientas a utilizarse

Para el diseño de la parte gráfica se utilizarán las siguientes herramientas:

Diseño gráfico

- **Blender-** Es un programa informático multiplataforma, dedicado especialmente al modelado, iluminación, renderizado, animación y creación de gráficos

tridimensionales. El programa fue inicialmente distribuido de forma gratuita, pero sin el código fuente, con un manual disponible para la venta, aunque posteriormente pasó a ser software libre. Actualmente es compatible con todas las versiones de Windows, Mac OS X, GNU/Linux (Incluyendo Android), Solaris, FreeBSD e IRIX.

- **Gimp**- Es un programa de edición de imágenes digitales en forma de mapa de bits, tanto dibujos como fotografías, es un programa libre y gratuito. GIMP tiene herramientas que se utilizan para el retoque y edición de imágenes, dibujo de formas libres, cambiar el tamaño, recortar, hacer fotomontajes, convertir a diferentes formatos de imagen, y otras tareas más especializadas. Se pueden también crear imágenes animadas en formato GIF (Graphics Interchange Format) e imágenes animadas en formato MPEG (Moving Picture Experts Group) usando un plugin de animación.

- **InkScape-** Es un editor de gráficos vectoriales gratuito y de código libre. InkScape puede crear y editar diagramas, líneas, gráficos, logotipos, e ilustraciones complejas. El formato principal que utiliza el programa es SVG (Scalable Vector Graphics).

- **Audacity**- Es una aplicación informática multiplataforma libre, que se puede usar para grabación y edición de audio, distribuido bajo la licencia GPL (General Public License).

Desarrollo

Para el desarrollo del videojuego se ha seleccionado Unity 3D como framework y motor de juego en base a un análisis y evaluación previa, además se han elegido las siguientes herramientas complementarias para cumplir con los objetivos del proyecto.

• **Unity 3D**.- Es un motor de videojuego multiplataforma creado por Unity Technologies. Unity está disponible como plataforma de desarrollo para

Microsoft Windows, OS X y Linux. La plataforma de desarrollo tiene soporte de compilación con diferentes tipos de plataformas. Desde su versión 5.4.0 ya no soporta el desarrollo de contenido para navegador a través de su plugin web en su lugar se utiliza WebGL. Unity tiene dos versiones: Unity Professional (pro) y Unity Personal.

• **Visual Studio 2015**.- Es un programa de desarrollo para sistemas operativos Windows desarrollado y distribuido por Microsoft Corporation.

Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros.

• **Android SDK**. - El SDK (Software Development Kit) de Android, incluye un conjunto de herramientas de desarrollo. Comprende un depurador de código, biblioteca, un simulador de teléfono basado en QEMU, documentación, ejemplos de código y tutoriales. El SDK es indispensable para poder compilar desde Unity hacia Android.

Roles del equipo

Para el desarrollo del videojuego se asignaron los siguientes roles mostrados en la Tabla 9

Persona	Contacto	Rol
Isidro García Hernández	isidrogarciahernandez@gmail.com	DAV Master
		Dueño del Producto
		Equipo DAV (Sonidista)
		Equipo DAV (Diseñador)
		Equipo DAV (Programador)

Tabla #9: Roles

Fuente: Isidro García y Manuel Galán

Historias de usuario

Las siguientes historias de usuario presentan las funcionalidades que tendrá el videojuego “Hello Remi”.

ID	Como	Quiero.../Quiero que...	De modo que.../Para
HU01	Jugador	Acceder al juego, salir, configurar y ver los créditos	Acceder al juego, salir, configurar y ver los créditos
HU02	Jugador	Antes de iniciar el juego ver la historia y las motivaciones del los personajes.	Motivar a empezar el juego y causar diversión.
HU03	Jugador	Tener la posibilidad de cambiar la calidad gráfica del juego	Mejorar el rendimiento del juego en caso de que el dispositivo no sea muy potente.
HU04	Jugador	Visualizar las instrucciones del juego por medio de algún tutorial	Entender la mecánica y controles del juego.
HU05	Jugador	Visualizar una pantalla de créditos.	Conocer quienes fueron los que desarrollaron el juego.

ID	Como	Quiero.../Quiero que...	De modo que.../Para
HU06	Jugador	Visualizar un menú para seleccionar los mundos y niveles.	Acceder a los mundos y niveles disponibles
HU07	Jugador	El personaje tenga diferentes atuendos o vestimentas.	Evitar la monotonía y tener mas variedad dentro del juego
HU08	Jugador	Visualizar una pantalla que muestre las puntuaciones obtenidas al finalizar el nivel.	Motivar al jugador ha rejugar el nivel para mejorar su puntaje.
HU09	Programador	Cada casilla del mapa se identifique con una coordenada única.	Facilitar el movimiento del personaje dentro del mapa
HU10	Programador	El personaje tenga diferentes acciones ejecutables.	Facilitar el desplazamiento y la interacción con el mundo.
HU11	Jugador	Existan enemigos variados dentro del mapa.	Ofrecer una mejor experiencia jugable con mayor retos para el jugador
HU12	Jugador	Un mapa con elevaciones y una estructura variada.	Brindar un mayor desafió a la hora de resolver el nivel
HU13	Jugador	La cámara posea una vista aérea y en tercera persona.	Se tenga una mejor visualización del tablero del juego desde diferentes perspectivas.
HU14	Programador	Tener comandos para ejecutar las acciones del personaje.	Facilitar la ejecución de las acciones por medio de un lenguaje común.
HU15	Diseñador	El personaje sea divertido y amigable.	Crear un personaje divertido y amigable para el jugador
HU16	Jugador	El juego tenga al menos dos idiomas para escoger.	Facilitar la comprensión del del juego a personas que no dominen el idioma español.
HU17	Jugador	El personaje posea acciones avanzadas y pueda tomar decisiones.	Mejorar la experiencia jugable con una mayor capacidad de estrategia dentro del juego.
HU18	Programador	Tener comandos avanzados para ejecutar las acciones avanzadas del personaje.	Facilitar la ejecución de las acciones por medio de un lenguaje común.
HU19	Diseñador	La decoración del mundo este acorde a los personajes y la historia del juego.	Crear un ambiente divertido para el jugador.
HU20	Jugador	El personaje principal cambie de animación según la acción que realice.	Crear un personaje con movimientos naturales y divertidos.

ID	Como	Quiero.../Quiero que...	De modo que.../Para
HU21	Programador	Controlar las colisiones del jugador con los objetos del mundo.	Crear realismo entre el personaje y los objetos del mundo, reaccionando de diferente manera según el objeto con el que se colisione.
HU22	Jugador	Visualizar una pantalla de pausa que permita reiniciar o salir de la partida.	El jugador pueda reiniciar o salir de la partida rápidamente.
HU23	Programador	Mostrar una notificación en la pantalla si los comandos se ingresan de forma incorrecta.	Evitar errores internos que impidan el correcto funcionamiento del juego.
HU24	Jugador	Visualizar los comandos que serán ejecutados.	Retroalimentar al usuario al visualizar los comandos que serán ejecutados.
HU25	Jugador	El personaje tenga powerUps para mejorar sus habilidades.	Mejorar la experiencia jugable y aumentar la diversión del juego.
HU26	Jugador	Visualizar la salida o meta del juego.	La meta permitirá dar por terminado el nivel del juego.
HU27	Jugador	El personaje pierda el nivel si sus vidas llegan a 0.	Aumentar la dificultad del juego y mejorar la experiencia.
HU28	Sonidista	Efectos de sonidos y música de fondo para el videojuego	Retroalimentar al jugador cuando escuche los sonidos de determinadas acciones.

Tabla #10: Historias de usuario del videojuego

Fuente: Isidro García y Manuel Galán

Pruebas de aceptación

Estas pruebas permiten validar las diferentes funcionalidades del videojuego una vez desarrolladas, de esta forma se garantizará que cumplan con lo requerido y pasarán al estado de completado.

Prueba de Aceptación	
Identificador: PA01	**Historia de Usuario (Nro. y Nombre):** HU01- Como jugador quiero visualizar el menú principal del videojuego
Nombre: Realizar el menú principal del juego	
Descripción: Crear una pantalla con botones que permitan entrar al juego, salir y cambiar las configuraciones del juego.	
Condiciones de ejecución: Debe ejecutarse en el dispositivo móvil, adaptarse a cualquier pantalla y funcionar correctamente.	
Entrada/Pasos de ejecución: - Presentar el menú principal con sus respectivos botones.	
Resultado esperado: Acceder a las respectivas pantallas según el botón seleccionado.	
Evaluación de prueba: Correcto	

Tabla #11: Prueba de aceptación “PA01”

Fuente: Isidro García y Manuel Galán

Reserva del producto

En la Tabla 12 se observan las historias de usuario con sus respectivas pruebas de aceptación (PA), puntos de prioridad (P) y estimación (E) según el dueño del producto.

P	Nro. HU	Ítem	E	Estado	PA
4	HU01	Como Jugador, quiero visualizar el menú principal del videojuego	1	Pendiente	PA01
4	HU02	Como jugador quiero antes de iniciar el juego ver la historia y las motivaciones del los personajes.	2	Pendiente	PA02
4	HU03	Como jugador quiero tener la posibilidad de cambiar la calidad gráfica del juego	2	Pendiente	PA03
4	HU04	Como jugador quiero visualizar las instrucciones del juego por medio de algún tutorial	1	Pendiente	PA04
4	HU05	Como jugador quiero visualizar una pantalla de créditos.	1	Pendiente	PA05
4	HU06	Como jugador quiero visualizar un menú para seleccionar los mundos y niveles.	1	Pendiente	PA06
4	HU07	Como jugador quiero que el personaje tenga diferentes atuendos o vestimentas.	2	Pendiente	PA07
4	HU08	Como jugador quiero visualizar una pantalla que muestre las puntuaciones obtenidas al finalizar el nivel.	2	Pendiente	PA08

P	Nro. HU	Ítem	E	Estado	PA
1	HU09	Como programador quiero que cada casilla del mapa se identifique con una coordenada única.	2	Pendiente	PA09
1	HU10	Como jugador quiero que el personaje tenga diferentes acciones ejecutables.	1	Pendiente	PA10
2	HU11	Como jugador quiero que existan enemigos variados dentro del mapa.	1	Pendiente	PA11
2	HU12	Como jugador quiero un mapa con elevaciones y una estructura variada..	1	Pendiente	PA12
2	HU13	Como jugador quiero que la cámara posea una vista aérea y en tercera persona.	2	Pendiente	PA13
1	HU14	Como programador quiero tener comandos para ejecutar las acciones del personaje.	2	Pendiente	PA14
1	HU15	Como jugador quiero que el personaje sea divertido y amigable.	1	Pendiente	PA15
4	HU16	Como jugador quiero que el juego tenga al menos dos idiomas para escoger	1	Pendiente	PA16
3	HU17	Como jugador quiero que el personaje posea acciones avanzadas y pueda tomar decisiones.	1	Pendiente	PA17
3	HU18	Como programador quiero tener comandos avanzados para ejecutar las acciones avanzadas del personaje.	1	Pendiente	PA18
3	HU19	Como diseñador quiero que la decoración del mundo este acorde a los personajes y la historia del juego.	1	Pendiente	PA19
1	HU20	Como diseñador quiero que el personaje principal cambie de animación según la acción que realice.	3	Pendiente	PA20
2	HU21	Como programador quiero controlar las colisiones del jugador con los objetos del mundo.	1	Pendiente	PA21
2	HU22	Como jugador quiero visualizar una pantalla de pausa que permita reiniciar o salir de la partida.	1	Pendiente	PA22
2	HU23	Como programador quiero mostrar una notificación en la pantalla si los comandos se ingresan de forma incorrecta.	2	Pendiente	PA23
1	HU24	Como jugador quiero visualizar los comandos que serán ejecutados.	1	Pendiente	PA24
3	HU25	Como jugador quiero que el personaje tenga powerUps para mejorar sus habilidades.	2	Pendiente	PA25
1	HU26	Como jugador quiero visualizar la meta o salida del juego	3	Pendiente	PA26

P	Nro. HU	Ítem	E	Estado	PA
3	HU27	Como jugador quiero que el personaje pierda el nivel si sus vidas llegan a 0.	1	Pendiente	PA27
3	HU28	Como sonidista quiero efectos de sonidos y música de fondo para el videojuego	1	Pendiente	PA28

Tabla #12: Reserva del producto del videojuego

Fuente: Isidro García y Manuel Galán

CAPÍTULO V

"**Conclusiones** y **Recomendaciones**", se establece las conclusiones donde llega el investigador de acuerdo a la solución planteada y desarrollada, también se define las recomendaciones con respecto a la aplicación.

7. CONCLUSIONES

- ✓ El uso de un framework y motor de juego apropiado ayudó a organizar de una mejor manera el desarrollo del proyecto, pero hay que destacar la complejidad existente al tratarse de un software muy diferente al que se ha venido desarrollando en las diferentes asignaturas de la carrera, se requiere de gran dedicación, tiempo y esfuerzo para integrar los diferentes elementos que componen el videojuego junto con las mejores prácticas de programación aprendidas.

- ✓ Dentro del desarrollo de videojuegos a nivel personal no existe una metodología ágil establecida formalmente, pero se han creado proyectos como la metodología DAV la cual se basa en SCRUM y XGD. Esta metodología permitió un desarrollo planificado y ordenado en cada una de las fases, añadiendo bastante flexibilidad al momento de realizar cambios en tareas que ya finalizaron. Las iteraciones ayudaron bastante a tener en corto tiempo una versión funcional del juego para analizar el estado del producto y las características que aún faltan desarrollar.

- ✓ La publicación del videojuego se la realizo sin inconvenientes, después de una evaluación entre diferentes tiendas online se eligió a PlayStore por la facilidad que brindan a los nuevos desarrolladores independientes. Gracias a las facilidades de la tienda se lograron muy buenos resultados tanto en críticas por parte de usuarios como en número de descargas después de realizar la publicación del videojuego.

8. RECOMENDACIONES

- ✓ El videojuego ha sido optimizado para un gran número de dispositivos Android, es recomendable revisar los requisitos de hardware y software necesarios que se encuentran en la página de publicación y en el manual de usuario.

- ✓ Antes de empezar una partida del juego es recomendable revisar la ayuda inicial del videojuego la cual brinda ayuda sobre los controles, comandos, errores, objetivos y todos los elementos que conforman el videojuego.

- ✓ Aunque dentro del proyecto el videojuego está limitado a un rango específico para estudiantes de primero y segundo año, el videojuego puede ser jugado por personas de cualquier edad por la simplicidad de los controles y la fácil comprensión del objetivo principal del videojuego. No es necesario tener ningún conocimiento sobre lógica de programación para jugarlo, ya que el contenido educativo se encuentra oculto dentro de la mecánica del juego.

9. REFERENCIAS BIBLIOGRAFICAS

- [1] SchoolControl.com, “La importancia de la programación en los pequeños,” 2015. [Online] Available: https://goo.gl/kbJkzk.
- [2] ElMundo.es, “Enseñar a niños a programar es lo más ’in’: Las mejores herramientas,” 2015. [Online] Available: https://goo.gl/II9pZA.
- [3] TheGuardian.com, “Coding at school: a parent’s guide to england’s new computing curriculum,” 2014. [Online] Availa-ble: https://www.theguardian.com/technology/2014/sep/04/coding-school-computing-children-programming.
- [4] Educacion.gob.ec, “El perfil de salida de los estudiantes de educación general básica,” 2015. [Online] Available: https://educacion.gob.ec/wp-content/uploads/downloads/2012/08/Perfil_Salida_EGB.pdf.
- [5] A. Rossaro, “Los videojuegos y su potencial educativo,” 2012. [Online] Available: https://goo.gl/ElyJFq.
- [6] Cbsnews.com, “Coding for kindergarteners: App teaches kids computer basics,” 2014. [Online] Available: https://goo.gl/C0KZhi.
- [7] J. Biggs, “Light-Bot Teaches Computer Science With A Cute Little Robot And Some Symbol-Based Programming,” 2015. [Online] Available: https://goo.gl/IRPkmR.
- [8] E. Cisneros, Videojuego Educativo como apoyo a la enseñanza de la Algoritmia para los estudiantes del Programa Nacional de Formación en Sistemas e Informática. PhD thesis, Instituto Superior Politécnico José Antonio Echeverría (Caracas), 2010.
- [9] K. Albuja, Análisis, diseño y desarrollo de un juego didáctico de razonamiento abstracto en 3d, para ayudar al desarrollo del pensamiento de niños entre 4 y 8 años, utilizando un game engine con c# y aplicando la metodología OOHDM. PhD thesis, 2012.
- [10] M. Zambrano, Diseño y desarrollo de un video-juego educativo con técnicas de inteligencia artificial para plataforma Android utilizando metodología OOHDM. Caso de estudio: Laberinto 3D. PhD thesis 2014.

- [11] M. Torres, Aplicación de la Metodología Oohdm y Técnicas de Inteligencia Artificial en la Solución del Desarrollo de un videojuego, enfocado a niños de 6 a 10 años, utilizando la Tecnología gdi+ basado en c# y Wiimote, para su aplicación en la Empresa Virtual Learni. PhD thesis, 2013.
- [12] C. T. Miller, Games: purpose and potential in education. Springer Science & Business Media, 2008.
- [13] D. V. Fernandez, C. M. Angelina, and EspaCursos, Desarrollo de Videojuegos: Arquitectura del Motor de Videojuegos. Cursos en Español, Oct. 2011.
- [14] Pixelsmil.com, “Qué significa "Indie",” 2012. [Online] Available: http://www.pixelsmil.com/2012/05/que-significa-indie.html.
- [15] J. P. Martín, “Penetración de los videojuegos educativos e infantiles en España desde el 2005 al 2007,” Comunicación y Pedagogía: Nuevas tecnologías y recursos didácticos, no. 229, pp. 23–28, 2008.
- [16] L. N. de Calidad del Software (España), “Metodologías y Ciclos de Vida,” 2009. [Online] Available: https://goo.gl/bkO7yO.
- [17] P. Kruchten, The rational unified process: an introduction. Addison-Wesley Professional, 2011.
- [18] Msdn.microsoft, “Microsoft Solutions Framework (MSF),” 2016. [Online] Available: https://msdn.microsoft.com/es-es/library/jj161047(v=vs.120).aspx.
- [19] O. Pastrana, “5 beneficios de aplicar metodologías ágiles en el desarrollo de software,” 2014. [Online] Available: https://goo.gl/wCMAvt.
- [20] N. Acerenza, A. Coppes, G. Mesa, A. Viera, E. Fernández, T. Laurenzo, and D. Vallespir, “Una Metodología para Desarrollo de Videojuegos,” 2009.
- [21] O. Trejos, La esencia de la Lógica de Programación. 2010.
- [22] E. Evans, Domain-Driven Design: Tackling Complexity in the Heart of Software. Addison Wesley, 2010.

.