PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

FACULTAD DE INGENIERÍA

ESCUELA DE SISTEMAS

DISERTACIÓN PREVIA A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS Y COMPUTACIÓN

GUÍA METODOLÓGICA PARA LA CREACIÓN DE VIDEOJUEGOS VINCULANDO METODOLOGÍAS DE DESARROLLO

MARCOS GYÖRGY SIERRA PALM

DIRECTOR: FABIÁN DE LA CRUZ DOMÍNGUEZ

QUITO, 2019

Agradecimientos

Quiero agradecer a mi familia por siempre estar pendientes de mí, proveerme con todo lo necesario para estudiar y vivir cómodamente, a mis profesores por enseñarme todos los conocimientos que han sido y serán de utilidad en el futuro, a mis amigos por apoyarme incondicionalmente y ser una compañía inolvidable, y al lector del trabajo por estar interesado en los temas que se tratan dentro del mismo. ¡Gracias!.

Contenido

Índice de Ilustraciones

Índice de tablas

INTRODUCCIÓN

Al elaborar videojuegos en la industria se siguen las metodologías ágiles, sin embargo estas son utilizadas para aplicaciones en general y no están enfocadas particularmente en los videojuegos. Además, la construcción de estos programas es, comúnmente, en equipo y, en consecuencia, requiere de diversos recursos y personal capacitado en diferentes campos para producir un software de calidad media a alta.

El propósito de este trabajo es elaborar una guía para el desarrollo de videojuegos desde una perspectiva individual, que pueda aplicarse en el transcurso del ciclo de vida de una metodología, seguirse con facilidad y no requiera de mayor conocimiento en programación. Actualmente se pueden encontrar una variedad de tutoriales y consejos en foros, blogs, páginas de soporte del software, entre otros, pero, la información está dividida, es difícil de captar para los principiantes y no presentan ejemplos claros de su uso.

El valor del presente trabajo será tener todos los pasos e información necesaria para desarrollar e implementar un videojuego siguiendo una metodología, de tal manera que sea simple de seguir mediante un ejemplo creado siguiendo la guía descrita.

El trabajo ha sido dividido en cinco capítulos para su desarrollo, los cuales están distribuidos de la siguiente manera:

Dentro del primer capítulo se define el concepto de un videojuego y se establecen las metodologías frecuentemente usadas en el desarrollo de videojuegos. En el segundo capítulo se presentan los entornos de desarrollo utilizados en diferentes géneros de la industria y sus características. El tercer capítulo trata sobre la familiarización de cada aspecto a usar en la elaboración del software, y en el cuarto capítulo se desarrolla un videojuego usando lo explicado anteriormente. Finalmente, en el quinto capítulo se presentan las conclusiones y recomendaciones de esta disertación.

CAPITULO 1: MARCO TEÓRICO

Dentro del capítulo 1 se presenta la justificación del trabajo mediante la descripción de la situación actual de la industria y la forma de trabajo de los equipos. Además, se establece el concepto de un videojuego en base a distintas perspectivas. Por último, se establecen las características principales de las metodologías Scrum y Extreme Programming (XP).

1.1 Justificación

Los videojuegos en su inicio tuvieron un desarrollo caracterizado principalmente por la simplicidad debido a las capacidades técnicas de la época. Pero, con los varios avances de la informática, evolucionó las capacidades tanto de software como hardware, además de la aplicación de metodologías y algoritmos. De esta manera en los últimos años la industria de los videojuegos ha crecido y cambiado hasta alcanzar niveles impensables en lo que se refiere a jugabilidad como también a calidad gráfica.

Esta evolución se encuentra ligada a distintos hitos que han afectado a la industria, algunos de los cuales han sido el uso de tecnología poligonal en 3D, la popularización de las computadoras personales como medios de multi-uso, el crecimiento del Internet, los motores de juegos, y actualmente, las redes sociales y el uso masivo de dispositivos móviles. (Fernandez & Angelina, 2011)

Y, como cualquier industria relacionada con el desarrollo de software, el uso de metodologías es esencial para garantizar la calidad y la producción estable del producto. Así, las metodologías ágiles como Scrum y XP, son las que se utilizan comúnmente por los equipos de trabajo. Sin embargo, si bien pueden ser aplicadas para la creación de videojuegos, no se enfocan específicamente en éstos y como consecuencia no prestan atención a características adicionales presentes en el ciclo de vida de aquellos.

Por lo tanto, se considera necesario la elaboración de una guía para el desarrollo de videojuegos aplicando una metodología centrada en el proceso de su creación, tanto desde la concepción hasta su liberación, con el propósito de facilitar y aclarar el uso de dicha metodología, además de la generación del videojuego.

1.2 Concepto de Videojuego

Desde un punto de vista general, un videojuego es asociado con el control que tiene el usuario sobre uno o varios personajes o entidades que buscan alcanzar diferentes objetivos en un mundo virtual. Este mundo a su vez tiende a estar compuesto por distintos escenarios en dos o tres dimensiones y se rige bajo una serie de reglas que establecen la interacción con el mismo. De esta forma, se presenta una relación entre el jugador y el videojuego, donde se definen retos con la finalidad de entretener y divertir al usuario. (Fernandez & Angelina, 2011)

De forma más formal se considera a los videojuegos como programas informáticos que son diseñados y desarrollados para el entretenimiento y diversión, a través de distintas tecnologías como las consolas de juego, las computadoras o los dispositivos móviles. (Juárez & Mombiela, 2011)

Por lo tanto, los videojuegos son programas o aplicaciones que mediante el uso de elementos como: personajes o entidades, música, escenarios y arte, narrativa y mundos virtuales con un conjunto de reglas definidas; presentan un medio por el cual el usuario es capaz de divertirse y superar retos establecidos por el juego.

1.3 Metodologías relacionadas: Scrum, Extreme Programming (XP)

Debido a la naturaleza volátil de los videojuegos, las metodologías utilizadas para su desarrollo se guían bajo principios ágiles. Esto se debe a que son iterativas, incrementales, interactúan de manera frecuente con el cliente y, ante todo, son flexibles considerando los requerimientos cambiantes. Otro punto que resaltar es que las decisiones son tomadas a base de la experiencia, sin seguir un proceso establecido ni técnicas particulares. Todo proyecto, en promedio, es realizado en equipos de tres a cinco personas, cada uno con rol distinto entre productor, programador, artista gráfico, diseñador de juego y artista sonoro.

El uso de metodologías agiles para el desarrollo de videojuegos se popularizó en los últimos años debido a varios casos de éxito en empresas que adaptaron las mismas. Entre los casos documentados, las

metodologías usadas con frecuencia son Scrum, XP o una combinación de ambas. Sin embargo, ninguna de estas adaptaciones está especificada formal y públicamente. (Acerenza, et.al., 2009)

1.3.1 Scrum

Scrum basa todas sus prácticas en un proceso incremental e iterativo. La metodología opera de esta manera: Al inicio de cada iteración, el equipo revisa que debe hacer y selecciona lo que a su criterio puede convertirse en un incremento potencial de funcionalidad al final de la iteración. Después se deja al equipo para que realice dicho incremento por el resto de la iteración y al final de ésta, el proyecto es inspeccionado, evaluado y ajustado, para realizar las adaptaciones necesarias a tiempo. (Schwaber, 2004)

Teniendo en cuenta el proceso con el que se guía Scrum inmediatamente se aprecia su utilidad en el desarrollo de videojuegos. Esto se debe a que los videojuegos, tanto los que se encuentran en desarrollo como los publicados, se encuentran en constante cambio a causa de la retroalimentación de los usuarios y el cambio de preferencias en la audiencia dirigida. Así, Scrum presenta una metodología que se enfoca en la adaptabilidad y el incremento de funcionalidad, características que son esenciales en la industria de los videojuegos. En la Ilustración 1 se presenta el funcionamiento general de SCRUM.

Ilustración 1 Esqueleto de Scrum (Schwaber, 2004)

1.3.2 Extreme Programming (XP)

XP puede ser descrito como una metodología que se enfoca en realizar solamente las actividades necesarias para generar valor para el cliente. Además, XP se adapta a requerimientos vagos o que

cambian rápidamente, adaptándose a las alteraciones espontaneas del mundo del negocio moderno. XP tiene cuatro valores principales:

- Comunicación: XP coloca a desarrolladores y cliente en comunicación constante. El cliente trabaja con los desarrolladores para establecer prioridades de negocio y contestar dudas del proyecto.
- Retroalimentación: La retroalimentación frecuente permite ejecutar ajustes constantes y aprender de ellos. El cliente es capaz de apreciar sus funcionalidades más importantes implementas tan rápidas como sean posibles.
- Simplicidad: Significa construir solo el sistema que realmente requiere ser construido, es decir, solucionar solamente los problemas actuales en el momento.
- Coraje: Hacer las decisiones difíciles cuando son necesarias. En otras palabras, si algo no funciona, solucionarlo. Si no cumple con la calidad establecida, mejorarlo. Si no se puede cumplir con las fechas de entrega, comunicarlo con el cliente. (Chromatic, 2003)

En la Ilustración 2 se aprecia el ciclo que sigue Extreme Programming.

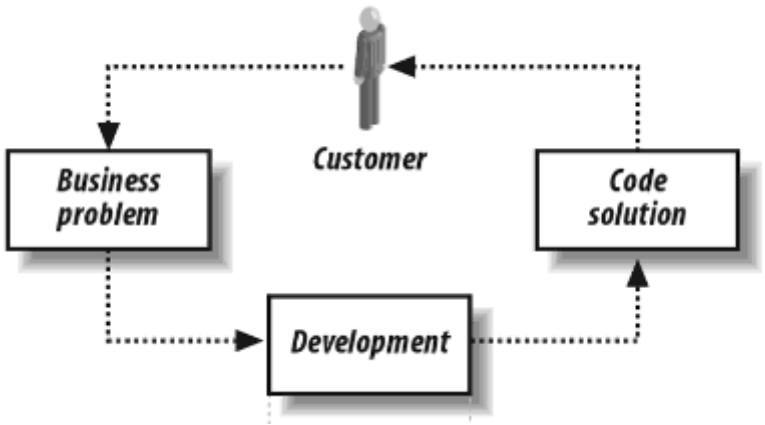


Ilustración 2 Ciclo XP (Chromatic, 2003)

Con estos conceptos establecidos, existen partes de esta metodología que resultan atractivas para el desarrollo de videojuegos, principalmente su enfoque en el cliente o usuario final y la situación actual. Debido a la naturaleza volátil de la industria, planear a futuro, a veces, no resulta útil ya que pueden

presentarse cambios repentinos que alteren las especificaciones establecidas, por lo tanto, es recomendable centrarse en lo actual y adaptarse a lo que ocurre en el transcurso del tiempo. Además, tomar en cuenta los deseos del usuario es de extrema importancia si el objetivo final es que el juego sea considerado como un producto de calidad.

CAPITULO 2: Plataformas para creación de videojuegos

Dentro del capítulo 2 se describen seis distintas herramientas para la creación de videojuegos a través de las características que se consideran relevantes a cada caso y la utilidad que representan dentro de la industria.

2.1 Unity 3D

Unity 3D es un motor de videojuegos multiplataforma que está disponible en los sistemas operativos Microsoft Windows, macOS y Linux. Las distintas características de Unity se presentan a continuación:

- **Licencia:** Unity cuenta con tres tipos de licencia, la "Pro" que está orientada a profesionales y estudios, la "Plus" para desarrolladores independientes y la "Personal" con un enfoque a principiantes. De las mencionadas la primera y segunda tienen un modelo de suscripción mensual y la tercera es gratis.
- **Herramientas de uso asociado:** El motor puede utilizarse junto con varias herramientas que están sincronizadas para que cualquier cambio realizado se actualice en toda instancia del objeto a través del proyecto sin que se requiera importarlo manualmente. Estas herramientas son: Blender, 3ds Max, Maya, Softimage, Modo, ZBrush, Cinema 4D, Cheetah3D, Adobe Photoshop, Adobe Fireworks y Allegorithmic Substance
- **Lenguaje de Programación:** La programación de videojuegos se realiza mediante scripts basados en Mono, que es la implementación open source de .NET Framework. Existen tres lenguajes que pueden usarse: UnityScript, C# y Boo.
- **Plataformas de Exportación:** Unity es compatible con más de 25 plataformas entre PC, móvil, consolas y otros. Algunas de las plataformas compatibles son: WebGL, Windows, SteamOS, macOS, GNU/Linux, iOS, Android, Windows Phone, PlayStation 4, Xbox 360, Xbox One, Wii U, Nintendo 3DS, Nintendo Switch, Oculus Rift, Google Cardboard, HTC Vive, PlayStation VR.
- **Tienda de Recursos:** Unity incluye una tienda de recursos con una variedad tanto gratis como pagada que facilita y agiliza el desarrollo de videojuegos. (Unity Technologies, 2019)

Las razones esenciales por las que Unity es uno de los motores de videojuegos más populares en la industria son su gran compatibilidad de plataformas, ya que puede ser compilado para otros sistemas operativos sin necesidad de mayores cambios. La tienda de recursos es útil tanto para principiantes como profesionales debido a su cantidad de opciones y el valor que aporta la actualización automática de objetos al modificarlos en las herramientas que pueden ser relacionadas con el motor.

Estas características además de una variedad de funciones adicionales vuelven a Unity, especialmente para desarrolladores nuevos, una de las opciones preferidas en lo que se refiere a aprendizaje y creación inicial de videojuegos.

2.2 Unreal Development Kit (Unreal Engine)

Unreal Development Kit aplicando la política "gratis para uso educacional y no comercial", permitía el uso de Unreal Engine 3 sin costos adicionales, sin embargo desde el año 2015 cuando la versión actual de Unreal Engine (Unreal Engine 4) se volvió disponible al público, se detuvo el soporte para Unreal Engine 3 y Unreal Development Kit. Por lo tanto, ya que no es necesario de una versión distinta para hacer uso de Unreal Engine 4, se considera a Unreal Engine como el equivalente de Unreal Development Kit.

Unreal Engine es un motor de videojuegos multiplataforma que está disponible en los sistemas operativos Microsoft Windows y macOS. Las características principales de Unreal Engine son las siguientes:

- **Licencia:** Unreal Engine 4 otorga una licencia no exclusiva, no transferible y no sublicenciable, para usar, reproducir, mostrar, ejecutar y modificarla tecnología, sin embargo en caso de que se lance al mercado algún producto y éste llegue a generar una ganancia mayor a $3000, entonces se debe pagar una regalía del 5% de los primeros $3000 generados por juego por trimestre.
- **Lenguaje de Programación:** El lenguaje utilizado por Unreal Engine es C++ y como complemento cuenta con "Blueprints" que permiten desarrollar juegos de forma simple y

visual. En esencia son similares a plantillas con funciones para diferentes elementos del videojuego que pueden ser editadas sin necesidad de codificar directamente.

- **Plataformas de Exportación:** Al ser multiplataforma es compatible con las plataformas más populares en la industria como son: Microsoft Windows, macOS, Linux, SteamOS, iOS, Android, PlayStation 4, Xbox One, Nintendo Switch, Oculus Rift, HTC Vive, PlayStation VR.
- **Unreal Engine Marketplace:** El Mercado de Unreal Engine contiene una amplia gama de recursos que varían desde personajes, ambientes, texturas hasta música, plugins e incluso "Blueprints". Estos pueden ser tanto gratis como pagados. (Epic Games, Inc, 2019)

A pesar de haber sido desarrollado inicialmente con la intención de ser usado principalmente en juegos del género First Person Shooter, es decir juegos donde la cámara tiene una perspectiva de primera persona y el jugador debe disparar con algún tipo de arma a enemigos u objetivos para acumular puntos, ha alcanzado gran popularidad en otros géneros y se encuentra como uno de los motores más utilizados para el desarrollo de videojuegos profesionales. Ya que su lenguaje de programación es C++, presenta una gran accesibilidad para los programadores, e incluso si el desarrollador principiante no tuviese un alto conocimiento del lenguaje, gracias a los "Blueprints" y el Marketplace la creación de un videojuego es relativamente sencilla.

2.3 Game Maker Studio

Similar a otros programas para desarrollo de videojuegos Game Maker Studio es multiplataforma y el IDE puede ser utilizado en Windows y macOS. Entre sus aspectos principales se encuentra:

- **Licencia:** Game Maker Studio es un software propietario con diferentes ofertas que permiten publicar juegos en diferentes plataformas e incluyen más recursos y funcionalidades. "Creator" permite publicar en PC, "Developer" en PC y móviles, y "Console" en las plataformas anteriores además de consolas. Sin embargo, cuenta con una versión de prueba ilimitada donde es posible familiarizarse con el diseño de videojuegos y el uso de la herramienta.
- **Lenguaje de Programación:** La aplicación tiene un diseño enfocado en usuarios sin conocimiento de programación ya que utiliza un sistema que permite arrastrar y soltar los

diferentes elementos del videojuego con acciones predefinidas lo que permite un desarrollo intuitivo para principiantes. Pero si se desea personalizar los juegos y aumentar funcionalidades, Game Maker incluye un lenguaje de programación con scripts conocido como Game Maker Language o GML, basado en el lenguaje C.

- **Plataformas de Exportación:** Los juegos hecho con Game Maker pueden ser exportados para las plataformas: Windows, Mac OS X, Ubuntu, Android, iOS, HTML5, PlayStation 4, and Xbox One.
- **Marketplace:** El Mercado de Game Maker contiene una variedad de recursos creados por la comunidad, pagados o gratuitos. (YoYo Games, 2019)

Una de las características que diferencia a Game Maker Studio de otros programas para desarrollar videojuegos es su facilidad de manejo y aprendizaje. Con una interfaz bastante intuitiva y sin la necesidad de conocer lenguajes de programación o poder programar, permite que cualquier usuario pueda crear su propio videojuego. Sin embargo, debido a sus restricciones en cuanto a la licencia y funcionalidades no es tan popular como otros en la industria. Aun con estos aspectos en consideración sigue siendo una gran herramienta para nuevos desarrolladores que desean aprender el diseño de videojuegos y puede ser usada como un escalón para después utilizar motores más profesionales.

2.4 M.U.G.E.N.

M.U.G.E.N. es uno de los motores de videojuegos más antiguos y que en la actualidad ya no tiene soporte, aun así, presenta características que se pueden encontrar en motores de videojuegos actuales y de esta forma se logra apreciar una evolución de estos programas basado en su funcionamiento en años anteriores.

Algunas de las características que se pueden encontrar en M.U.G.E.N. son:

- **Licencia:** M.U.G.E.N. inicialmente tenía una licencia definida como "M.U.G.E.N. is free for non-commercial use", que significa que puede ser utilizado mientras que no tenga fines comerciales.

Sin embargo, debido a que Elecbyte, la empresa responsable de M.U.G.E.N. no renovó los permisos para los usuarios y éstos manteniendo su uso bajo el pretexto de que era una versión BETA, el software cambió de ser shareware a freeware, siendo actualmente totalmente gratis.

- **Lenguaje de Programación:** M.U.G.E.N. no utiliza un lenguaje de programación para poder crear videojuegos, esto se debe a que sigue un modelo similar al de Game Maker en el que se enfoca en usuarios sin conocimientos de programación y les provee de todos los elementos necesarios para desarrollar un videojuego sin necesidad de codificar algún objeto.
- **Plataformas de Exportación:** Debido a su antigüedad las únicas plataformas con las que es compatible M.U.G.E.N. son las de escritorio, es decir: Microsoft Windows, Linux, macOS y MS-DOS. (Elecbyte, 2019)

El caso de M.U.G.E.N. no es raro de encontrar en los motores de videojuegos, si bien es cierto que los motores modernos tienen un mayor enfoque al desarrollo profesional y variado de videojuegos, todavía existen varios motores que tienen un enfoque o se especializan en un género particular. Aunque M.U.G.E.N. cuenta con todos los elementos necesarios para crear un videojuego, estos solo pueden ser del género de lucha, que se refiere a juegos donde el jugador selecciona un personaje o un equipo y se enfrenta a otro jugador con su respectiva selección durante un periodo de tiempo y rondas definidas, y el que gane más rondas es el que consigue la victoria.

Esta especialización puede considerarse como una desventaja en cuanto a variedad, pero la ventaja de motores enfocados en un solo género es que facilitan y agilizan el proceso de desarrollo cuando se desea crear un juego de un tema solamente. Por lo cual se sacrifican recursos por rapidez y sencillez de trabajo.

2.5 RPG Maker

RPG Maker es una serie de programas que siguen el modelo de poder crear videojuegos a pesar de no tener conocimientos de programación. Este modelo ha permitido que se mantenga como uno de los motores más populares en la industria a pesar de la versatilidad que presentan otras aplicaciones relativamente nuevas.

Entre las características que presenta el motor, las más notables son:

- **Licencia:** RPG Maker cuenta con diversas versiones del software que se diferencian ya sea por calidad y cantidad de los recursos, aumento de funciones o variedad en los géneros que pueden ser desarrollados. Todas las versiones son de tipo propietario, pero permiten ser descargadas bajo una versión de prueba.
- **Lenguaje de Programación:** Debido al modelo que sigue RPG Maker no es necesario de programación por parte del usuario para ser capaz de crear videojuegos. Sin embargo, ya que utiliza scripts para el funcionamiento de los objetos, en caso de que el usuario pueda programar, este puede hacerlo si desea personalizar su juego. Los lenguajes varían dependiendo de la versión que se utiliza entre Ruby, C++, Java, Javascript o HTML5.
- **Plataformas de Exportación:** La compatibilidad de los juegos creados en RPG Maker difiere por versión usada, sin embargo, las versiones más recientes exportan para Microsoft Windows, MAC, Linux, dispositivos móviles, Playstation 4, Nintendo DS, Nintendo 3DS, Nintendo Switch y web. (RPG Maker, 2019)

RPG Maker es uno de los motores que más tiempo lleva en funcionamiento en la industria y a pesar de aquello mantiene su posición dentro de la misma. Sus versiones principalmente pueden ser usadas solamente en PC, pero, existen versiones que se utilizan en consolas y mediante las cuales se puede compartir directamente lo desarrollado con la comunidad.

Como se aprecia en el nombre, RPG Maker es un motor que está especializado en la creación de videojuegos del género “Role Playing Game”, es decir juegos de rol donde el propósito principal es que el jugador experimente una aventura controlando al protagonista e interactuando con el entorno y otros personajes, similar a una película pero con la influencia del usuario. Y aunque actualmente es posible el desarrollo de juegos de otro género, la razón principal de su uso es para juegos donde la historia es el factor predominante. Aún con estas restricciones sigue siendo de los programas más utilizados demostrando el atractivo de que cualquiera puede presentar su historia a través de un videojuego.

2.6 Ren'Py

El último motor por considerar es Ren'Py el cual cumple una función particular en la industria por su enfoque en el género de “Novelas Visuales” que para algunos no son considerados videojuegos. Esto se debe a que el género es similar a leer una novela pero con la adición de escenarios y personajes estáticos, a veces con animaciones, voces de los personajes y sonido ambiental, donde el jugador es el protagonista de la historia y en ciertos puntos elije opciones que influyen en el final que tendrá el juego. Pero, ya que el género cumple con los requisitos mínimos para ser considerado un videojuego, se juzga pertinente mencionarlo.

El motor puede ser ejecutado en los sistemas operativos Windows, macOS y Linux, y cuenta con las siguientes características:

- **Licencia:** A diferencia de otros motores de videojuegos, Ren'Py es completamente libre de uso y sus juegos pueden ser comercializados sin pagar regalías, sin embargo porciones de Ren'Py son derivadas de código que es licenciado bajo la GNU Lesser General Public License o LGPL, por lo cual los juegos hechos con Ren'Py deben ser distribuidos de manera que satisfaga la LGPL.
- **Lenguaje de Programación:** El programa incluye funciones básicas para el desarrollo de una historia, por ejemplo: ramificación de historia, transición de escenarios, puntos de guardado, entre otros y para usuarios más avanzados cuenta con scripts hechos en lenguaje Python mediante los cuales se puede añadir más opciones.
- **Plataformas de Exportación:** Ren'Py permite desarrollar videojuegos compatibles con las plataformas Android, Chrome OS, Linux, Windows, macOS y iOS. (Ren'Py, 2019)

Las novelas visuales eran géneros que solo tenían popularidad en Asia, principalmente Japón, pero han ganado un espacio en la industria global, razón por la cual motores de videojuegos enfocados en ellas han sido desarrollados. Debido a la simplicidad de las novelas visuales, tanto en su desarrollo como en su jugabilidad, es comprensible porque se prefiere el uso de un programa especializado en esta área en comparación con uno de gran potencia con una alta versatilidad pero que requiere un mayor conocimiento y esfuerzo para una aplicación que no lo necesita.

CAPITULO 3: Creación de la guía metodológica propuesta

En el capítulo 3 se detalla el proceso y flujo de trabajo que se seguirá para la creación del videojuego de ejemplo. Además, se definen los roles que se identifican dentro de la industria y las actividades que se ejecutan en cada fase del proceso. Por último, se establecen las herramientas a utilizar para la implementación del proyecto.

3.1 Flujo

Previo a la presentación del flujo de trabajo propuesto, se presenta los ciclos de vida de las metodologías ágiles utilizadas como base para la elaboración del proceso de trabajo planteado. Las metodologías que se mencionan son Scrum y Extreme Programming (XP).

3.1.1 Ciclo de Vida Scrum

El ciclo de vida de Scrum contiene 5 fases que se describen brevemente a continuación:

1. Planificación del Sprint: En esta fase se reúne todo el Equipo de Scrum para planificar las actividades a realizar durante el Sprint, el incremento que éste representa en el proyecto y el objetivo final del Sprint.
2. Scrum Diario: Se realiza diariamente con el objetivo de que el equipo de desarrollo tenga definido las tareas que se deben efectuar durante la jornada de trabajo de aquel día. Utiliza los resultados obtenidos del anterior Scrum Diario como base para estimar el avance que se puede alcanzar durante las siguientes 24 horas.
3. Desarrollo: Durante este periodo es indispensable que no se desvíe del objetivo planteado y es posible alterar el alcance del Sprint con la aprobación del Propietario del Producto en caso de que se presenten inconvenientes imprevistos que puedan afectar el tiempo de entrega o el incremento planificado.
4. Revisión del Sprint: En la revisión, todos los participantes se reúnen de forma informal para discutir los resultados del Sprint, lo que estuvo bien hecho, los problemas que se encontraron, como se resolvieron, el estado actual del producto, entre otros aspectos, con el objetivo de

determinar posibles mejoras en futuros Sprints o alterar características del producto si el propietario lo considera adecuado.

5. Retrospectiva del Sprint: La retrospectiva, similar a la revisión, tiene el propósito de identificar mejoras para futuros Sprints pero con un enfoque en la forma de trabajo del Equipo de Scrum y las herramientas, procesos y relaciones involucradas. Una forma de mejorar futuros Sprints es analizando las historias entregadas en relación a las planificadas y cuantas incidencias se encontraron durante el Sprint, así se puede ajustar la cantidad de historias seleccionadas para la siguiente iteración con el propósito de aumentar la velocidad del equipo y la calidad de lo desarrollado.

3.1.2 Ciclo de Vida XP

El ciclo de vida de XP consta de 6 fases:

1. Exploración: En esta fase el cliente define las historias de usuario, que son equivalentes a los requerimientos, y el equipo de desarrollo prueba las herramientas y tecnologías que serán usadas en el proyecto con el objetivo de analizar la viabilidad de la arquitectura planteada.
2. Planificación de la Entrega: El cliente establece la importancia de cada historia y el equipo de desarrollo estima el tiempo necesario para cada una de ellas. Se definen la historia o conjunto de historias contenidas en la primera entrega y se genera un cronograma de trabajo.
3. Iteraciones: Esta es la fase de desarrollo de XP donde al final de todas las iteraciones planificadas anteriormente se consigue un producto listo para ser implementado en un ambiente laboral.
4. Producción: Las actividades a realizar durante este periodo son pruebas del sistema, tanto de las funcionalidades como el rendimiento para asegurar que cumpla con lo establecido y que se traslade sin inconvenientes al entorno del cliente.
5. Mantenimiento: Ya que el sistema que se generó al final de las iteraciones no siempre es el producto completo o se desea añadir funcionalidades al sistema en el futuro, es necesario de un

equipo que se encargue de mantener en funcionamiento el sistema actual durante el desarrollo de nuevas versiones.

6. Muerte del Proyecto: Esta es la finalización del proyecto y sucede cuando se ha cumplido con los requisitos del cliente, tanto en funcionalidades desarrolladas como el rendimiento esperado. Sin embargo, también se puede dar cuando las versiones iniciales no generan las ganancias esperadas o si el presupuesto no es suficiente para seguir con la producción.

3.2 Elaboración de la guía metodológica propuesta

3.2.1 Roles

Es necesario también definir roles que participan en el proceso para clasificar las actividades a ser ejecutadas y los responsables de éstas. Los roles propuestos son los siguientes:

Líder del Proyecto: Es la persona encargada de guiar a el(los) equipo(s) de trabajo para asegurar que cumplan con los objetivos asignados y participar en la solución de los problemas que se presentan en el transcurso del proyecto.

Equipo de Desarrollo: Es similar a los equipos que se encuentran en el desarrollo de software, sin embargo, a diferencia de aquellos equipos que están conformados principalmente de programadores e ingenieros de software, el Equipo de Desarrollo en la industria de videojuegos incluye individuos o grupos de trabajo formados en disciplinas distintas a la programación.

Esta diferencia amerita una división en el Equipo de Desarrollo en subroles que son:

- Programador: Este cumple con las tareas esenciales en el desarrollo de software como son la definición de la arquitectura, el diseño, la implementación del videojuego junto con la unión del contenido audiovisual producido por los otros subroles.
- Diseñador Gráfico: Encargado del producir el arte de los personajes y escenario, diseño 2D y 3D, animación, entre otras características artísticas.

- Diseñador de Sonido: Responsable de todo aquello relacionado con los efectos de sonido y banda sonora que se puede lograr mediante la grabación de sonidos originales o el uso y manipulación de existentes.
- Diseñador de Jugabilidad: Las actividades que se asignan a este subrol son diseñar la forma en cómo se jugará, el concepto de los personajes, el entorno en el que se desenvuelve el jugador, los niveles y retos que se deben afrontar, es decir la experiencia del usuario.

Cliente: Es la entidad que aporta con los intereses y deseos de los usuarios finales, necesarios para adaptar el producto durante su desarrollo. Es indispensable para asegurar que el videojuego cumpla con las expectativas del mercado y sea un éxito.

Beta Tester: El rol que cumple es el de jugar el producto en sus últimas etapas de desarrollo y verificar que la funcionalidad cumpla con los requerimientos establecidos con anterioridad. Una característica a tomar en cuenta es que puede tener o no conocimientos en videojuegos o ser un jugador habitual, y aun así participar en las pruebas del producto, ya que es importante todo tipo de resultados y respuestas para mejorar la experiencia.

3.2.2 Ciclo de Vida Propuesto

Tomando en cuenta los Ciclos de Vida de las metodologías mencionadas, se toman como base algunos de sus aspectos y se los adapta a la realidad de los entornos de trabajo en la industria de los videojuegos para elaborar la visión macro del Ciclo de Vida que se observa en la Ilustración 3.

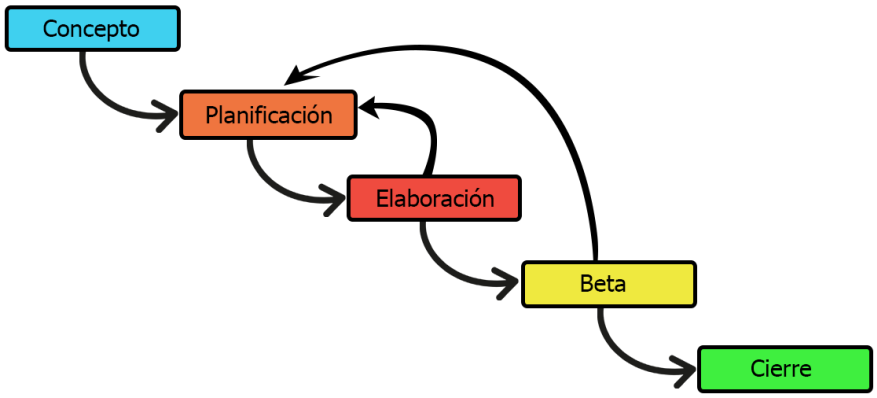


Ilustración 3 Ciclo de Vida – Visión Macro (Marcos Sierra, 2019)

3.2.3 Fase de Concepto

Esta fase se refiere a la definición de requerimientos y mediante ellos el establecimiento del concepto del videojuego. Este concepto no tiene que contar con todos los aspectos requeridos y basta con tener una idea general de lo que se va a desarrollar ya que se concretará con cada iteración, en base a los resultados obtenidos y la retroalimentación del cliente.

En consideración de estos detalles, se espera que la fase de concepto tenga una duración máxima de 2 días y consta de pocas actividades a realizar. Las actividades principales que se consideran son:

- **Definir los aspectos de monetización:** Se establece el público objetivo y como se obtendrán ingresos, es decir mediante micro transacciones, la venta del juego completo, modelo gratis, pero con publicidad, entre otros.
- **Definir las características principales del juego:** Se discute el ambiente en el que se desenvuelven los personajes, el tipo de juego, la jugabilidad que mejor se adapte al videojuego y otros aspectos necesarios para empezar el desarrollo. Una práctica común en la industria para asegurar que se mantenga la calidad esperada y que el producto final sea entretenido, es definir pruebas periódicas mediante prototipos para que los beneficiarios sean capaces de dar sus opiniones y alterar el funcionamiento acorde a lo mencionado.

En la Ilustración 4 se aprecia el diagrama de actividades de la fase de Concepto y el Apéndice A. contiene el formato del entregable de la Fase de Concepto.

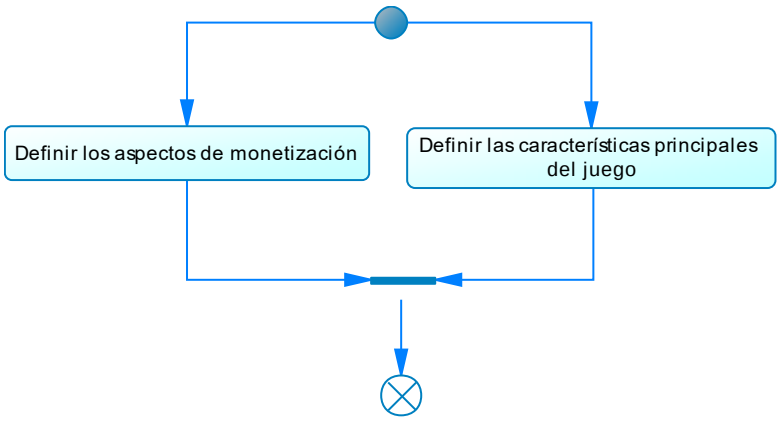


Ilustración 4 Actividades - Fase de Concepto (Marcos Sierra, 2019)

3.2.4 Fase de Planificación

Como se aprecia en otras metodologías, la planificación es un componente esencial para asegurar que el resto de las fases y actividades se ejecuten con tiempo y en caso de que se presenten inconvenientes, lograr manejarlos sin aumentar el costo total del proyecto. Ya que la planificación es propensa a cambiar para adaptarse a la situación actual del proyecto, no tendrá un modelo demasiado rígido y por lo tanto la duración de la fase consta de 1 a 3 días, donde se definen los puntos clave.

Las actividades principales para desarrollar son:

- **Definir objetivos:** Se definen los objetivos y resultados que se esperan obtener al finalizar el proyecto, y los objetivos específicos de la iteración actual. Estos son necesarios para guiar el desarrollo en la dirección correcta.
- **Definir el cronograma de trabajo:** Se establecen las fechas y puntos de control de las actividades restantes en base al concepto del juego y los riesgos estimados. Se identifican también la cantidad de iteraciones necesarias para completar el proyecto. Esta estimación se realiza mediante experiencias de proyectos anteriores y los análisis realizados en los postmortem de aquellos proyectos. En casos donde no se cuente con referencias anteriores, se recomienda estimar el tiempo en una reunión con el equipo de trabajo, donde cada uno de los integrantes analizan cuanto tiempo se considera necesario para la finalización de las actividades planteadas y se define una aproximación del análisis hecho.
- **Definir presupuesto:** Se definen los recursos económicos necesarios para el proyecto, además del personal necesario para el Equipo de Desarrollo.
- **Especificación de Requerimientos:** En esta actividad se determinan las funcionalidades con las que cuenta el videojuego, en que plataformas puede ejecutarse, los requisitos técnicos, se estima el tiempo necesario para la elaboración de cada una y se define el orden de desarrollo en base a la prioridad que representa para el proyecto.

En la Ilustración 5 se aprecia el diagrama de actividades de la fase de Planificación. Además, el Apéndice B. contiene el formato del entregable de la Fase de Planificación y el Apéndice C. incluye el formato de la Especificación de Requerimientos.

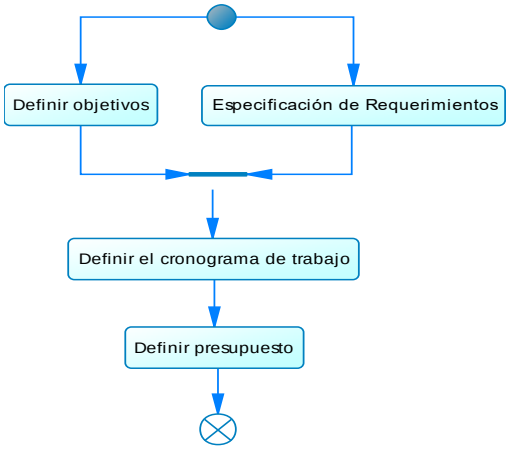


Ilustración 5 Actividades - Fase de Planificación (Marcos Sierra, 2019)

3.2.5 Fase de Elaboración

Como es de esperar de todo proyecto, la fase de elaboración es la que con frecuencia es la más larga en completar. Lo que se busca durante esta fase es la implementación del videojuego mediante un modelo iterativo e incremental, y generando cada cierto periodo de tiempo un prototipo para obtener la aprobación del cliente o cualquier cambio que desee efectuar en caso de estar inconforme, siempre tomando en cuenta los plazos y tiempos planificados.

Las actividades representativas de la fase son:

- **Definir aspectos de la Iteración:** Dentro de cada Iteración es necesario establecer los objetivos o metas que se pretende alcanzar al finalizarla. Estos objetivos sirven a su vez para controlar el desarrollo adecuado de lo planteado.

  Para elegir que funcionalidades se desarrollan en la Iteración se basa en la planificación establecida y la prioridad definida, tomando en cuenta que la duración total no supere el de la Iteración. Además, las funcionalidades se dividen en actividades en correspondencia con las áreas especializadas en ellas con el propósito de facilitar la verificación y validación de lo elaborado, y una mayor productividad.

- **Desarrollo de la funcionalidad:** Una vez escogida la funcionalidad, cada miembro del Equipo de Desarrollo selecciona las actividades relacionadas con su área de experticia y se encarga de su elaboración guiado por los objetivos definidos.
- **Control de la Iteración:** Con el propósito de asegurar el correcto cumplimiento de las actividades es necesario de un monitoreo constante que pueda identificar problemas emergentes, y a partir de éstos divisar posibles soluciones. Este monitoreo también permite comunicar al Cliente el avance del proyecto y dar a consideración al Equipo de Desarrollo como se encuentran en relación con lo cumplido por el resto de las áreas.
- **Evaluación de los resultados:** Al finalizar la Iteración se valida la funcionalidad generada, tanto por parte del Cliente como por el Equipo de Desarrollo, y se ajusta el plan de trabajo en base a la retroalimentación obtenida. Los cambios que se pueden presentar son en la prioridad de las actividades, creación, modificación o eliminación de funcionalidades, y de no ser necesario mayores cambios, se analizan los errores y defectos encontrados durante el periodo de desarrollo y se elaboran formas de afrontarlas para el futuro.

Para poder realizar un control de la iteración y que todo el equipo este consciente del progreso del trabajo se utiliza un tablero de control como se aprecia en la Ilustración 6.

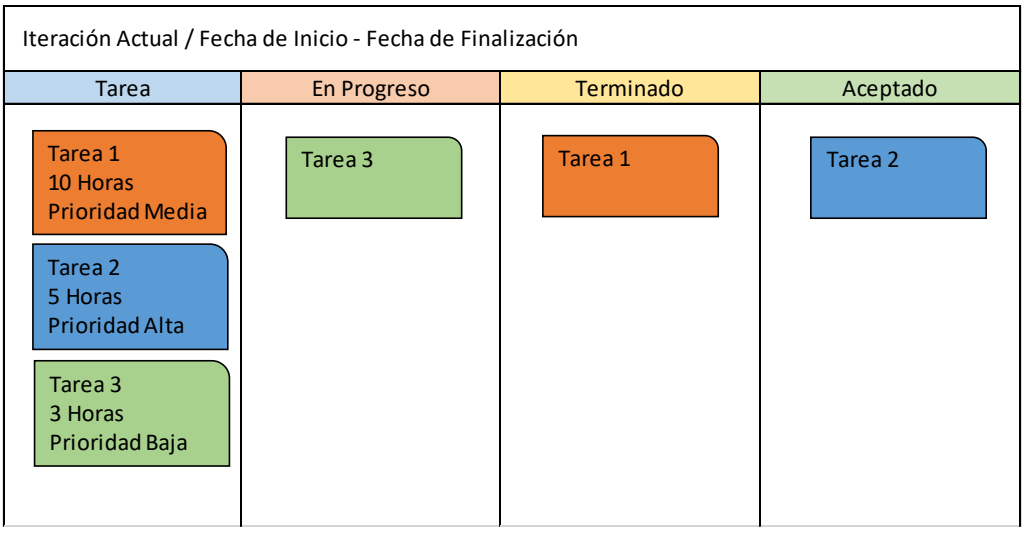


Ilustración 6 Tablero de Control (Marcos Sierra, 2019)

La duración de cada iteración puede variar dependiendo de la dificultad de las funcionalidades que se implementan o también de la cantidad que se planificó desarrollar, sin embargo, lo ideal es conseguir

un avance en no más de 2 semanas con el fin de ofrecer al Cliente una visualización del progreso constante.

En la Ilustración 7 se aprecia el diagrama de actividades de la fase de Elaboración.

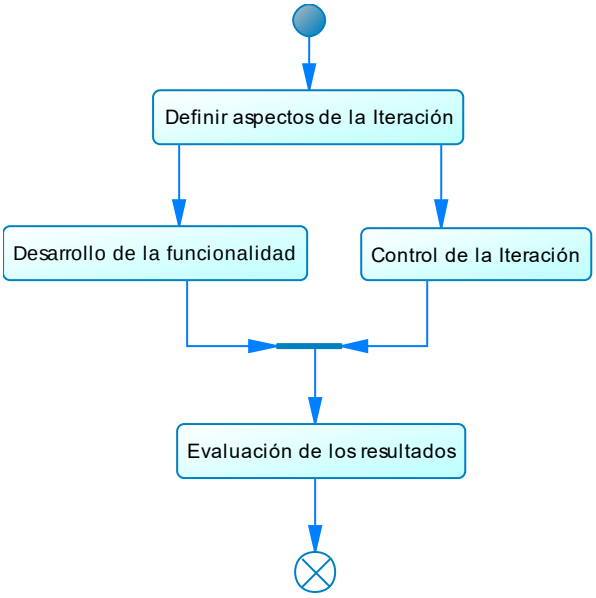


Ilustración 7 Actividades - Fase de Elaboración (Marcos Sierra, 2019)

3.2.6 Fase de Beta

Durante esta fase se realizan pruebas de los prototipos generados en los periodos definidos en la planificación, con el propósito de modificar las funcionalidades para que mejoren su factor de entretenimiento o solucionar problemas informados por el grupo de Beta Testers seleccionados.

El tiempo que se destina para la ejecución de las pruebas Beta no debe superar una semana, ya que es suficientemente largo para encontrar los defectos más importantes y una duración mayor representa una pérdida de tiempo.

En caso de que las pruebas reflejen que la situación actual del proyecto requiere de cambios radicales, se considera aceptable el cambio del cronograma, presupuesto, aumento o reemplazo de integrantes del Equipo de Desarrollo, entre otras modificaciones que permitan lograr el éxito del proyecto.

Las actividades que conforman la fase son:

- **Ejecución de pruebas:** Se determina los medios por los cuales se distribuyen las versiones Beta del videojuego, la cantidad de Beta Testers a utilizar, y los puntos clave a verificar. Una vez establecido estos parámetros, se entregan los prototipos a los usuarios y un documento con preguntas que permitan evaluar el videojuego, junto con un espacio para observaciones no definidas en el formulario.
- **Análisis de respuestas:** En base a las respuestas obtenidas se ajusta las características del videojuego para solucionar los errores que se presenten o pulir los aspectos que recibieron una buena acogida pero que no cumplen en su totalidad con los objetivos propuestos.

En la Ilustración 8 se aprecia el diagrama de actividades de la fase Beta.

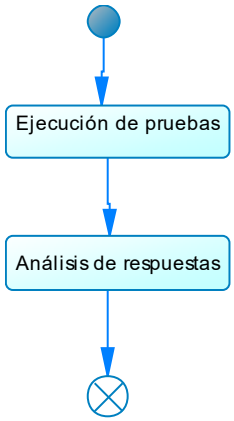


Ilustración 8 Actividades - Fase Beta (Marcos Sierra, 2019)

3.2.7 Fase de Cierre

En la última fase se genera el entregable final que se presentará al Cliente junto con la documentación relacionada y se realiza un postmortem que evalúa el rendimiento demostrado durante el desarrollo del proyecto con la finalidad de identificar en que aspectos tuvo éxito el proyecto, en que falló, como se solucionaron los problemas y que características pueden ser reutilizables para futuros proyectos.

En la Ilustración 9 se aprecia el diagrama de actividades de la fase de Cierre y el Apéndice D. contiene el formato utilizado para realizar el proceso de postmortem.

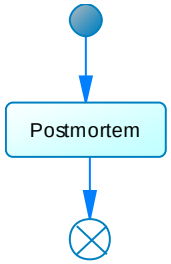


Ilustración 9 Actividades - Fase de Cierre (Marcos Sierra, 2019)

3.2.8 Responsables en cada fase

En la Tabla 1 se utiliza el modelo RACI, también conocido como una matriz de asignación de responsabilidades, para establecer los roles responsables de cada actividad en las distintas fases. Responsable es aquel que está encargado de realizar la tarea, Aprobador se asegura de que la actividad se haya realizado correctamente, Consultado son a quienes se les pide la opinión acerca de la actividad e Informado es a quienes se les mantiene al tanto de las actividades realizadas.

		ROLES			
FASE	ACTIVIDAD	Lider del Proyecto	Equipo de Desarrollo	Cliente	Beta Tester
Concepto	Definir los aspectos de comercialización	Responsable		Responsable	
	Definir las características principales del juego	Responsable		Responsable	
Planificación	Definir objetivos	Responsable	Responsable	Aprobador	
	Definir el cronograma de trabajo	Responsable	Responsable	Informado	
	Definir presupuesto	Responsable		Consultado	
	Especificación de Requerimientos	Responsable		Consultado	
Elaboración	Definir aspectos de la Iteración	Responsable	Responsable		
	Desarrollo de la funcionalidad	Consultado	Responsable		
	Control de la Iteración	Aprobador	Responsable		
	Evaluación de los resultados	Responsable	Responsable	Informado	
Beta	Ejecución de pruebas	Responsable			Responsable
	Análisis de respuestas	Responsable	Responsable	Consultado	Consultado
Cierre	Postmortem	Aprobador	Responsable	Informado	

Tabla 1 Roles Responsables (Marcos Sierra, 2019)

3.3 Plataforma y herramientas seleccionadas

Para el desarrollo del videojuego con el que se ejemplifica la guía y proceso propuesto se decide utilizar el motor de videojuegos Unity 3D debido a que es la herramienta que mejor se adapta a lo que se planea elaborar. Una de las razones es sus requerimientos mínimos de hardware y software los cuales son listados en la página oficial de descarga como:

OS: Windows 7 SP1+, 8, 10, 64-bit versions only; macOS 10.11+

GPU: Graphics card with DX10 (shader model 4.0) capabilities.

Otra característica por la cual es la herramienta preferida de uso es su amplia comunidad y documentación, necesaria en caso de que se presenten problemas y errores de los que se desconoce la solución. Por último, ya que también se utilizará Adobe Photoshop para el diseño artístico del juego y con la funcionalidad de Unity 3D que permite el enlazamiento directo de esta herramienta a su motor, el proceso de desarrollo se agiliza y facilita los cambios necesarios.

Para el aspecto visual la herramienta tentativa, en caso de ser necesario para el desarrollo, será Adobe Photoshop para diseño de imágenes. Ya se explicó anteriormente una de las ventajas de utilizar Adobe Photoshop, pero además de esta característica, también cuenta con una amplia gama de funcionalidades para la creación de componentes gráficos.

La plataforma objetivo para la que se planea desarrollar el proyecto es el sistema operativo Android principalmente por la facilidad de instalación de aplicaciones externas, con una opción tentativa a Windows en PC si se considera necesario y posible en el cronograma planificado. Esto se debe a la adaptabilidad que presenta Unity 3D cuando se desea exportar el proyecto a múltiples plataformas.

CAPITULO 4: Creación de un videojuego usando la guía metodológica propuesta

En el capítulo 4 se presenta el desarrollo de un videojuego usando las características planteadas anteriormente en el documento. La elaboración del videojuego tiene el fin de demostrar el uso de los formatos entregables y los resultados que se esperan conseguir en las distintas fases en un ambiente de trabajo real. Con este objetivo en consideración, se define al equipo de trabajo como un grupo nuevo a la industria aprendiendo el proceso relacionado con la creación de un videojuego por primera vez.

Para una mejor comprensión del contenido de este capítulo se recomienda revisar los Anexos correspondientes durante la lectura de cada fase, ubicados en la parte final de la disertación.

Se inicia describiendo de manera general el videojuego que se plantea desarrollar, después se planifican tres iteraciones para su creación completa, durante la cual se sigue la metodología propuesta y se destacan las características principales del proceso, como se demuestra a continuación.

4.1 Fase de Concepto

En la fase de Concepto se utiliza el Apéndice A, para poder documentar y establecer con claridad los aspectos esenciales del videojuego, esto con el fin de que en las siguientes fases se tenga un núcleo concreto en el que se pueda basar el resto de integrantes del proyecto para la planificación y desarrollo futuro.

A continuación se muestra el entregable generado para el videojuego a crear como ejemplo:

- Inicio del Ejemplo -

Modelo de Monetización

Debido a la naturaleza del videojuego a desarrollarse, se decide que el juego será gratis de descargar sin generar ingresos, ya que el propósito del producto es que el equipo de trabajo se familiarice con el proceso de desarrollo y la dinámica dentro de la industria de videojuegos Indie, que se refiere a juegos usualmente creados por individuos o equipos de desarrollo pequeños sin el soporte financiero de un

publicador grande de videojuegos, para que los siguientes proyectos se ejecuten con mayor agilidad y mejores resultados.

Características del Videojuego

Categorías	**Detalle**
Narrativa	No Aplica.
Cantidad de Jugadores	Uno.
Online	Sin necesidad.
Cooperativo o Competitivo	No Aplica.
Género(s)	Juego rítmico toca notas.
Temática	No Aplica.

Descripción General

El videojuego que se plantea es uno rítmico en el que el jugador debe presionar los objetivos en movimiento vertical hacia abajo que se presentan en pantalla en el momento adecuado, donde serán destruidos para generar una puntuación final. Los puntos pueden variar dependiendo de cuando fueron presionados.

Se espera que el jugador elija su propia música para jugar con el propósito de que cada persona pueda escuchar lo que desee mientras se entretiene. Debido a esto no se plantea que el juego sea competitivo u online, ya que cada persona tiene diferentes canciones y solo se espera crear un videojuego móvil que se pueda utilizar para pasar el tiempo durante el día.

- Fin del Ejemplo -

Como se observa el entregable generado incluye los aspectos esenciales necesarios para empezar la planificación del desarrollo y como se espera que se use el producto por los usuarios. Es necesario

establecer que el rol de Líder del Proyecto también representará el rol del Cliente, ya que es un proyecto con fines de aprendizaje y se encargará de dar la retroalimentación que provendría de los intereses del Cliente, necesaria para que el equipo de trabajo pueda continuar con la elaboración del producto.

4.2 Fase de Planificación

En la fase de Planificación se utiliza el Apéndice B, con el fin de establecer un cronograma de trabajo, objetivos generales y específicos del proyecto, y el Apéndice C, para detallar las características del videojuego dentro de la Especificación de Requerimientos (ER). Durante la elaboración del videojuego, se añadieron funcionalidades que inicialmente no fueron planificadas, por lo cual se presentaran varios documentos en los que se observan cambios en la planificación, además de distintas ER debido a que se genera uno por iteración.

A continuación se presentan los documentos generados durante la fase de Planificación detallando la iteración en la que fueron generados y los cambios realizados:

- Iteración 1 -

Objetivos Generales

- Desarrollar un videojuego móvil de calidad que pueda ser utilizado en cualquier momento.
- Aprender el proceso de desarrollo de un videojuego y la dinámica de la industria.
- Conocer los diferentes formatos entregables utilizados en el proyecto.

Objetivos Específicos de la Iteración

- Elaborar un prototipo inicial con la funcionalidad esencial del videojuego.
- Realizar una prueba de funcionamiento e identificar posibles mejoras y/o errores.
- Obtener la retroalimentación del Cliente para continuar con el proyecto o modificar características no aprobadas.

Cronograma de Trabajo

Funcionalidad	Actividades	Área Responsable	Requisitos Previos	Prioridad	Fecha Inicio	Fecha Fin Planeada	Fecha Fin Real
F1. Base del Juego	A1.1 Desactivación de Objetos	Desarrollo		Alta	09/09/2019	10/09/2019	
	A1.2 Movimiento de Objetos	Desarrollo		Media	09/09/2019	11/09/2019	
	A1.3 Registro de Puntos	Desarrollo	A1.1	Baja	10/09/2019	12/09/2019	
	A1.4 Reproducción de Audio	Desarrollo		Alta	09/09/2019	13/09/2019	
F2. Menús	A2.1 Inicio y Salida del Juego	Desarrollo		Baja			
	A2.2 Submenú de Selección de Música	Desarrollo	A2.1	Alta			
	A2.3 Submenú de Puntuación Final	Desarrollo	F1	Media			
F3. Diseño Artístico	A3.1 Sprites y Efectos de Objetos	Diseño	F1	Alta			
	A3.2 Diseño de Fondo de Nivel	Diseño y Desarrollo	F1	Media			
	A3.3 Sprites y Efectos de Botones	Diseño	F2	Baja			
	A3.4 Diseño de Fondo de Menús	Diseño	F2	Baja			

En el Apéndice E, se puede ver el ER para la iteración 1. En la iteración 1 se observa una estimación inicial de las funcionalidades necesarias para el funcionamiento correcto del juego, empezando el proceso por la funcionalidad más importante. En las siguientes iteraciones se actualizarán las fechas de inicio de las siguientes funcionalidades y las fechas finales reales.

- Fin Iteración 1 –

- Iteración 2 -

Objetivos Generales

- Desarrollar un videojuego móvil de calidad que pueda ser utilizado en cualquier momento.
- Aprender el proceso de desarrollo de un videojuego y la dinámica de la industria.
- Conocer los diferentes formatos entregables utilizados en el proyecto.

Objetivos Específicos de la Iteración

- Elaborar los menús necesarios para el flujo de escenas en el juego.
- Realizar una prueba de funcionamiento e identificar posibles mejoras y/o errores.
- Obtener la retroalimentación del Cliente para continuar con el proyecto o modificar características no aprobadas.

Cronograma de Trabajo

Funcionalidad	Actividades	Área Responsable	Requisitos Previos	Prioridad	Fecha Inicio	Fecha Fin Planeada	Fecha Fin Real
F1. Base del Juego	A1.1 Desactivación de Objetos	Desarrollo		Alta	09/09/2019	10/09/2019	10/09/2019
	A1.2 Movimiento de Objetos	Desarrollo		Media	09/09/2019	11/09/2019	10/09/2019
	A1.3 Registro de Puntos	Desarrollo	A1.1	Baja	10/09/2019	12/09/2019	11/09/2019
	A1.4 Reproducción de Audio	Desarrollo		Alta	09/09/2019	13/09/2019	12/09/2019
F2. Menús	A2.1 Inicio y Salida del Juego	Desarrollo		Baja	16/09/2019	17/09/2019	
	A2.2 Submenú de Selección de Música	Desarrollo	A2.1	Alta	17/09/2019	18/09/2019	
	A2.3 Submenú de Puntuación Final	Desarrollo	F1	Media	16/09/2019	17/09/2019	
F3. Diseño Artístico	A3.1 Sprites y Efectos de Objetos	Diseño	F1	Alta			
	A3.2 Diseño de Fondo de Nivel	Diseño y Desarrollo	F1	Media			
	A3.3 Sprites y Efectos de Botones	Diseño	F2	Baja			
	A3.4 Diseño de Fondo de Menús	Diseño	F2	Baja			

En el Apéndice F, se puede ver el ER para la iteración 2. En la iteración 2 se continúa con la siguiente funcionalidad en rango de importancia debido a que se consiguió la aprobación del cliente en la anterior iteración, sin embargo dentro de la fase de pruebas de la segunda funcionalidad el cliente presenta mejoras al juego las cuales se verán reflejadas en la planificación de la iteración 3.

- Fin Iteración 2 –

- Iteración 3 -

Objetivos Generales

- Desarrollar un videojuego móvil de calidad que pueda ser utilizado en cualquier momento.
- Aprender el proceso de desarrollo de un videojuego y la dinámica de la industria.
- Conocer los diferentes formatos entregables utilizados en el proyecto.

Objetivos Específicos de la Iteración

- Elaborar los diseños artísticos necesarios para los distintos objetos del juego.
- Agregar el menú de pausa al juego y la opción de seleccionar la dificultad del nivel.
- Realizar una prueba de funcionamiento e identificar posibles mejoras y/o errores.
- Obtener la retroalimentación del Cliente para continuar con el proyecto o modificar características no aprobadas.

Cronograma de Trabajo

Funcionalidad	Actividades	Área Responsable	Requisitos Previos	Prioridad	Fecha Inicio	Fecha Fin Planeada	Fecha Fin Real
F1. Base del Juego	A1.1 Desactivación de Objetos	Desarrollo		Alta	09/09/2019	10/09/2019	10/09/2019
	A1.2 Movimiento de Objetos	Desarrollo		Media	09/09/2019	11/09/2019	10/09/2019
	A1.3 Registro de Puntos	Desarrollo	A1.1	Baja	10/09/2019	12/09/2019	11/09/2019
	A1.4 Reproducción de Audio	Desarrollo		Alta	09/09/2019	13/09/2019	12/09/2019
F2. Menús	A2.1 Inicio y Salida del Juego	Desarrollo		Baja	16/09/2019	17/09/2019	17/09/2019
	A2.2 Submenú de Selección de Música	Desarrollo	A2.1	Alta	17/09/2019	18/09/2019	19/09/2019
	A2.3 Submenú de Puntuación Final	Desarrollo	F1	Media	16/09/2019	17/09/2019	17/09/2019
	A2.4 Menú de Pausa y Opción de Dificultad	Desarrollo	A2.2	Alta	20/09/2019	24/09/2019	
F3. Diseño Artístico	A3.1 Sprites y Efectos de Objetos	Diseño	F1	Alta	20/09/2019	23/09/2019	
	A3.2 Diseño de Fondo de Nivel	Diseño y Desarrollo	F1	Media	20/09/2019	23/09/2019	
	A3.3 Sprites y Efectos de Botones	Diseño	F2	Baja	20/09/2019	23/09/2019	
	A3.4 Diseño de Fondo de Menús	Diseño	F2	Baja	20/09/2019	23/09/2019	

En el Apéndice G, se puede ver el ER para la iteración 3. En la iteración 3 se finaliza el proyecto añadiendo a la planificación la actividad “A2.4 Menú de Pausa y Opción de Dificultad” especificada por el cliente. Con esto concluye las diferentes iteraciones en la fase de planificación las cuales estarán también presentes en las fases de Elaboración y Beta.

- Fin Iteración 3 –

Para terminar con la fase de Planificación se presenta el cronograma de trabajo completo con todas las fechas actualizadas al finalizar la tercera iteración.

Cronograma de Trabajo Final

Funcionalidad	Actividades	Área Responsable	Requisitos Previos	Prioridad	Fecha Inicio	Fecha Fin Planeada	Fecha Fin Real
F1. Base del Juego	A1.1 Desactivación de Objetos	Desarrollo		Alta	09/09/2019	10/09/2019	10/09/2019
	A1.2 Movimiento de Objetos	Desarrollo		Media	09/09/2019	11/09/2019	10/09/2019
	A1.3 Registro de Puntos	Desarrollo	A1.1	Baja	10/09/2019	12/09/2019	11/09/2019
	A1.4 Reproducción de Audio	Desarrollo		Alta	09/09/2019	13/09/2019	12/09/2019
F2. Menús	A2.1 Inicio y Salida del Juego	Desarrollo		Baja	16/09/2019	17/09/2019	17/09/2019
	A2.2 Submenú de Selección de Música	Desarrollo	A2.1	Alta	17/09/2019	18/09/2019	19/09/2019
	A2.3 Submenú de Puntuación Final	Desarrollo	F1	Media	16/09/2019	17/09/2019	17/09/2019
	A2.4 Menú de Pausa y Opción de Dificultad	Desarrollo	A2.2	Alta	20/09/2019	24/09/2019	25/09/2019
F3. Diseño Artístico	A3.1 Sprites y Efectos de Objetos	Diseño	F1	Alta	20/09/2019	23/09/2019	23/09/2019
	A3.2 Diseño de Fondo de Nivel	Diseño y Desarrollo	F1	Media	20/09/2019	23/09/2019	23/09/2019
	A3.3 Sprites y Efectos de Botones	Diseño	F2	Baja	20/09/2019	23/09/2019	23/09/2019
	A3.4 Diseño de Fondo de Menús	Diseño	F2	Baja	20/09/2019	23/09/2019	23/09/2019

4.3 Fase de Elaboración

Como se observó en la fase de Planificación durante la duración del proyecto se presentaron tres iteraciones en total, una por cada funcionalidad y debido a posibles mejoras que se encontraron en la fase Beta de cada iteración, fue necesario añadir una actividad extra a la funcionalidad F2. Menús que fue desarrollada en la tercera iteración.

La fase de Elaboración dentro del ciclo de vida de un proyecto es en la cual el equipo empieza a programar, diseñar el arte de los objetos y/o elaborar las pistas de audio planificadas para la(s) funcionalidad(es) de determinada iteración. Por lo tanto, no hay un documento que sea necesario completar ya que los entregables son los prototipos del juego o partes separadas de él para mostrar al Cliente el avance realizado.

Así, para ejemplificar está fase se presentan a continuación capturas del avance hecho en cada iteración, además de fragmentos de código necesario para desarrollar las varias actividades planificadas y los estándares de desarrollo utilizados que se encuentra en el Apéndice H:

- Iteración 1 –

Actividad A1.1 Desactivación de Objetos

La desactivación de objetos se refiere al momento en el que el jugador presiona la nota en movimiento, y es necesaria para que no se acumulen objetos que ocupan memoria innecesaria y pueden llegar a ralentizar la velocidad de respuesta del juego. También incluye la generación de los objetos que el jugador debe presionar.

En la Ilustración 10 se muestra el código en el cual se detecta la posición en la que fue presionada la pantalla, se transforma esa posición a coordenadas dentro del juego y se compara con la posición del objeto para saber si está dentro del rango aceptable para ser desactivado.

```
if (Input.anyKeyDown)
{
    Vector3 posMouse = Input.mousePosition;

    Vector3 posCamara = Camera.main.ScreenToWorldPoint(posMouse);

    posMouse.z = - (posCamara.z + 1);

    posMouse = Camera.main.ScreenToWorldPoint(posMouse);

    float posicionX = Mathf.Abs(posMouse.x - (transform.position.x));
    float posicionY = Mathf.Abs(posMouse.y - (transform.position.y));

    if (_puedePresionar && posicionX <= 2 && posicionY <= 2)
    {
        gameObject.SetActive(false);
    }
}
```

Ilustración 10 Código para detectar y desactivar el objeto presionado (Marcos Sierra, 2019)

En la Ilustración 11 se encuentra el código que define cuantos objetos deben ser generados en una cierta cantidad de tiempo, la posición en el eje X y el eje Y para que no colisionen entre ellos y el registro de cuantos objetos fueron creados para poder hacer el cálculo de puntaje a posterior.

```
IEnumerator GenerarEnemigo()
{
    yield return new WaitForSeconds(segundosSpawn);
    float limiteX = 0, limiteY = 0;
    List<float> limTempX = new List<float>();
    List<float> limTempY = new List<float>();
    bool condDistancia = true;
    int limiteSpawn = 1;

    if(limiteEnemigo != 1)
        limiteSpawn = Random.Range(1, 3);

    for (int i = 0; i < limiteSpawn; i++)
    {
        condDistancia = true;
        while (condDistancia)
        {
            limiteX = Random.Range(-22, 23);
            for (int j = 0; j < limTempX.Count; j++)
            {
                if (Mathf.Abs(limTempX[j] - (limiteX)) <= 4)
                {
                    condDistancia = false;
                    break;
                }
            }
            if (condDistancia)
            {
                condDistancia = false;
            }
            else
                condDistancia = true;
        }
```

```
            condDistancia = true;
            while (condDistancia)
            {
                limiteY = Random.Range(25, 31);
                int z = 0;
                for (int j = 0; j < limTempY.Count; j++)
                {
                    if (limTempY[j] == limiteY)
                    {
                        z++;
                        if (z > 2)
                        {
                            condDistancia = false;
                            break;
                        }
                    }
                }
                if (condDistancia)
                {
                    condDistancia = false;
                }
                else
                    condDistancia = true;
            }
            GameObject go = Instantiate(enemigo);
            go.transform.position = new Vector3(limiteX, limiteY, -1);
            //Agrega uno al numero de notas totales generadas
            GameManager._instancia.ConteoNotas();

            limTempX.Add(limiteX);
            limTempY.Add(limiteY);
        }
        isSpawning = false;
    }
```

Ilustración 11 Código para generar objetos (Marcos Sierra, 2019)

Actividad A1.2 Movimiento de Objetos

El movimiento de objetos define la velocidad y dirección en la que se traslada los objetos generados, además de añadir una función de destrucción del objeto en el momento que sale de los límites permitidos definidos. En la Ilustración 12 se presenta el código que compara la posición actual del objeto en el eje Y con el límite establecido, fuera del cual el objeto debe ser destruido. En caso de que el objeto siga dentro del rango aceptable seguirá moviéndose hacia abajo verticalmente hasta que sea presionado o salga de pantalla. Para el movimiento se utiliza el tiempo interno del dispositivo para que se traslade a una velocidad constante y de forma fluida.

```
if(_iniciado)
{
    if (transform.position.y >= _limiteInf)
    {
        transform.position -= new Vector3(0f, _velocidadNota * Time.deltaTime, 0f);
    }
    else if(transform.position.y <= _limiteInf && transform.position.x != 40)
    {
        Destroy(gameObject);
    }
}
```

Ilustración 12 Código para mover el objeto y destruirlo fuera de pantalla (Marcos Sierra, 2019)

Actividad A1.3 Registro de Puntos

El registro de puntos permite determinar los distintos puntajes que pueden ser obtenidos, además de contar las notas que no fueron presionadas a tiempo, para calcular una puntuación final. Para esto es necesario utilizar la desactivación de objetos para detectar la posición exacta en la que fue presionado un objeto y que tipo de puntuación equivale aquella posición.

En la Ilustración 13 se presenta el código que fue agregado a la función de destrucción a través de la cual se determina que puntaje obtiene el jugador dependiendo de las coordenadas en las que fue presionado el objeto. Una vez determinado se suma al puntaje total actual los puntos recibidos, multiplicando el bono actual por el valor relacionado con la posición presionada que puede ser Normal, Bueno, Perfecto, y en caso de fallar se reinicia el bono y no se altera la puntuación total.

```
if (_puedePresionar && posicionX <= 2 && posicionY <= 2)
{
    gameObject.SetActive(false);

    //Si nota Y es menor a -7.5 y mayor a -8.8, o menor a -11.2
    if((transform.position.y < (_limiteInf.transform.position.y + 2.5) &&
    transform.position.y > (_limiteInf.transform.position.y + 1.2)) ||
    transform.position.y < (_limiteInf.transform.position.y - 1.2))
    {
        GameManager._instancia.HitNormal();
        Instantiate(_efectoHit, transform.position, _efectoHit.transform.rotation);

    }//Si nota Y es menor a -8.8 y mayor a -9.5, o menor a -10.5 y mayor a -11.2
    else if((transform.position.y < (_limiteInf.transform.position.y + 1.2) &&
    transform.position.y > (_limiteInf.transform.position.y + 0.5)) ||
    (transform.position.y < (_limiteInf.transform.position.y - 0.5) &&
    transform.position.y > (_limiteInf.transform.position.y - 1.2)))
    {
        GameManager._instancia.HitBueno();
        Instantiate(_efectoGood, transform.position, _efectoGood.transform.rotation);

    }//Si nota Y es menor a -9.5 y mayor a -10.5
    else if(transform.position.y < (_limiteInf.transform.position.y + 0.5) &&
    transform.position.y > (_limiteInf.transform.position.y - 0.5))
    {
        GameManager._instancia.HitPerfecto();
        Instantiate(_efectoPerf, transform.position, _efectoPerf.transform.rotation);


    }
}
     public void NotaHit()
     {
         if(_bonusActual - 1 < _limitesBonus.Length)
         {
             _seguidorBonus++;

             if(_limitesBonus[_bonusActual - 1] <= _seguidorBonus)
             {
                 _seguidorBonus = 0;
                 _bonusActual++;
             }
         }
         _txtPunt.text = "Puntos: " + _puntActual;
         _txtBonus.text = "Bonus: x" + _bonusActual;
     }

     public void HitNormal()
     {
         _puntActual += _puntPorNota * _bonusActual;
         NotaHit();

         _hitsNormales++;
     }
```

Ilustración 13 Código para registrar el puntaje correspondiente (Marcos Sierra, 2019)

Actividad A1.4 Reproducción de Audio

La reproducción de audio se refiere a como se logra que determinada pista de audio suene en un momento específico dentro del juego. En el caso del presente ejemplo solo es necesario que se reproduzca la canción seleccionada por el usuario al iniciar el nivel, lo cual se puede lograr con el código que se encuentra en la Ilustración 14. Se define una variable tipo AudioSource necesaria para almacenar el audio y una vez que se define el audio deseado se lo reproduce mediante la función Play( ). Sin embargo, esto solo se aplica a la funcionalidad de reproducir sonido, y la dificultad radica en que se requiere reproducir la canción escogida por el jugador, y para aquello el código se encuentra en la segunda iteración al desarrollar el submenú de selección de música.

```
public AudioSource _musica;

public void setMusica()
{
    _musica = gameObject.AddComponent<AudioSource>();
    _musica.Play();
}
```

Ilustración 14 Código para reproducir audio (Marcos Sierra, 2019)

Esto concluye la Iteración 1 de la fase de Elaboración en donde se desarrolló la funcionalidad principal del juego que incluye las partes esenciales para la correcta ejecución del producto. Se mostraron los fragmentos de código que se consideraron pertinentes para explicar el funcionamiento, pero están presentes más componentes que, aunque sean de menor importancia, también afectan el comportamiento del software. Aun así, aquellas líneas de código sirven únicamente el propósito de complementar a los bloques principales que son presentados en las ilustraciones.

- Fin Iteración 1 –

- Iteración 2 -

A2.1 Inicio y Salida del Juego

El menú de inicio y salida del juego es el menú principal del juego que de la forma más básica sirve únicamente para empezar el juego o salir de él. En la Ilustración 15 se muestra el código utilizado para lograr dichas funciones, donde la primera función permite cambiar de escenas dentro del juego, las cuales son los equivalentes a las pantallas que tiene un programa, y la segunda función cierra la aplicación inmediatamente.

```
public void PlayGame()
{
    SceneManager.LoadScene("GameScene");
}

public void SalirGame()
{
    Application.Quit();
}
```

Ilustración 15 Código para iniciar o salir del juego (Marcos Sierra, 2019)

En la Ilustración 16 se presenta el menú principal inicial sin fondo, sombras o colores, creado únicamente por su funcionalidad.

Ilustración 16 Menú principal inicial (Marcos Sierra, 2019)

A2.2 Submenú de Selección de Música

El submenú de selección de música como se había mencionado antes es donde radica la dificultad de poder reproducir las canciones que el usuario elija para jugar ya que se requiere de una manera de acceder a la memoria del dispositivo, seleccionar y cargar el audio dentro del juego. En la Ilustración 17 se muestra el código para poder seleccionar la ruta en la cual se encuentra la canción deseada, para lo cual se utiliza un recurso gratis de la tienda de recursos de la comunidad que provee Unity llamado "Simple File Browser", que presenta un diseño similar a los DialogBox. Para utilizarla primero se importa la librería, después se definen al iniciar el juego los filtros de búsqueda deseados. Se relaciona al botón de seleccionar canción la función SelectSong( ), la cual inicia una corrutina que abre el DialogBox y retorna la ruta de la canción, que finalmente se envía como argumento a otra función para cargar el audio en la variable determinada.

```
using SimpleFileBrowser;
using TMPro;


    void Start()
    {
        FileBrowser.SetFilters(true, new FileBrowser.Filter("Music", ".mp3", ".wav"));
        FileBrowser.SetDefaultFilter(".mp3");
    }

    public void SelectSong()
    {
        StartCoroutine(ShowLoadDialog());
    }

    IEnumerator ShowLoadDialog()
    {
        yield return FileBrowser.WaitForLoadDialog(false, null, "Load File", "Load");

        Debug.Log(FileBrowser.Success + " " + FileBrowser.Result);

        GameManager._instancia.setMusica(FileBrowser.Result);
    }
```

Ilustración 17 Código para obtener la ruta de la canción seleccionada (Marcos Sierra, 2019)

En la Ilustración 18 continua el proceso de carga de audio donde se utiliza la ruta obtenida utilizando la clase WWW de Unity que permite recuperar contenido a través de URLs, así una vez cargado el

audio se comprueba si no existen errores y si no hay inconvenientes se establece la canción como el audio clip de la variable a ser reproducida en el nivel.

```
public void setMusica(string filename)
{
    WWW www = new WWW("file:///" + filename);

    if(www.error != null)
    {
        _txtError.text = "ARCHIVO NO SOPORTADO, INTENTE CON OTRO";
    }
    else
    {
        _condicionMusica = true;

        _musica = gameObject.AddComponent<AudioSource>();
        _musica.clip = www.GetAudioClip(false, true);
        _laMusica.origen = _musica;
        _txtError.text = "";
    }
}
```

Ilustración 18 Código para cargar la canción seleccionada (Marcos Sierra, 2019)

En la Ilustración 19 se presenta el submenú de selección de música inicial sin Sprites creado únicamente por su funcionalidad.

Ilustración 19 Submenú de selección de música inicial (Marcos Sierra, 2019)

A2.3 Submenú de Puntuación Final

El submenú de puntuación final es el que aparece al finalizar el nivel que presenta los resultados obtenidos como la cantidad de notas normales, buenas, perfectas acertadas, notas fallidas, el porcentaje acertado y un rango que se define desde “F” como lo peor hasta “SS” como el mejor rango posible. En la Ilustración 20 se aprecia la forma en la que se muestra por pantalla la puntuación obtenida y los criterios para obtener los distintos rangos.

```
_pantallaResult.SetActive(true);
_normalText.text = _hitsNormales.ToString();
_buenoText.text = _hitsBuenos.ToString();
_perfText.text = _hitsPerf.ToString();
_missText.text = _hitsMiss.ToString();

float totalAciertos = _hitsNormales + _hitsBuenos + _hitsPerf;
float porcentajeAciertos = (totalAciertos / _totalNotas) * 100f;

_porcentajeHitText.text = porcentajeAciertos.ToString("F1") + "%";

string rangoValor = "F";

if(porcentajeAciertos > 40)
{
    rangoValor = "D";
    if(porcentajeAciertos > 55)
    {
        rangoValor = "C";
        if(porcentajeAciertos > 70)
        {
            rangoValor = "B";
            if(porcentajeAciertos > 85)
            {
                rangoValor = "A";
                if(porcentajeAciertos > 95)
                {
                    rangoValor = "S";
                    if(porcentajeAciertos == 100)
                        rangoValor = "SS";
                }
            }
        }
    }
}
_rangoText.text = rangoValor;

_puntFinalText.text = _puntActual.ToString();
```

Ilustración 20 Código para determinar el puntaje final (Marcos Sierra, 2019)

En la Ilustración 21 se presenta el submenú de puntuación final.

Ilustración 21 Submenú de puntuación final inicial (Marcos Sierra, 2019)

Esta actividad concluye la Iteración 2 en la cual el enfoque fue en los menús considerados como indispensables para el funcionamiento del juego. Ya que en la Iteración 3 se planea actividades de diseño artístico los menús tienen un aspecto simple y robusto.

- Fin Iteración 2 –

- Iteración 3 -

A2.4 Menú de Pausa y Selección de Dificultad

El menú de pausa y la selección de dificultad fueron adiciones hechas al juego al notar posibles mejoras en la fase Beta, y tienen que ver con la opción de detener temporalmente el juego en caso de que se deban realizar otras actividades o si el usuario desea regresar al menú principal. La selección de dificultad en cambio permite elegir que tan desafiante desea el jugador que sea el nivel que está por jugar. En la Ilustración 22 se aprecia el código que permite pausar los elementos del juego y reanudarlos cuando el usuario lo desee. El código detiene la actualización del movimiento de los objetos y pausa la música.

```
public void Pausar()
{
    _pauseMenuUI.SetActive(true);
    Time.timeScale = 0f;
    GameManager._instancia.PausarNivel();
}

public void PausarNivel()
{
    _musica.Pause();
    _nivelPausado = true;
    _laMusica.empezarJuego = false;
}
```

Ilustración 22 Código para pausar el juego (Marcos Sierra, 2019)

En la Ilustración 23 se presenta el código que altera los valores del juego que definen la velocidad con la que se mueven los objetos y la cantidad de objetos que pueden generarse en un rango de tiempo, dependiendo de la dificultad que se escoja.

```
public void SeleccionarDificultad(int valor)
{
    if(valor == 1)
    {
        GameManager._instancia._elMovimiento._velocidadNota = 15;
        GameManager._instancia._laMusica.limiteEnemigo = 2;
        GameManager._instancia._laMusica.segundosSpawn = 0.8f;
    }
    else if(valor == 2)
    {
        GameManager._instancia._elMovimiento._velocidadNota = 25;
        GameManager._instancia._laMusica.limiteEnemigo = 1;
        GameManager._instancia._laMusica.segundosSpawn = 0.5f;
    }
    else if(valor == 3)
    {
        GameManager._instancia._elMovimiento._velocidadNota = 20;
        GameManager._instancia._laMusica.limiteEnemigo = 2;
        GameManager._instancia._laMusica.segundosSpawn = 0.5f;
    }
    _condDificultad = true;
    _txtError.text = "";
}
```

Ilustración 23 Código para definir la dificultad (Marcos Sierra, 2019)

En la Ilustración 24 se presenta el menú de pausa.

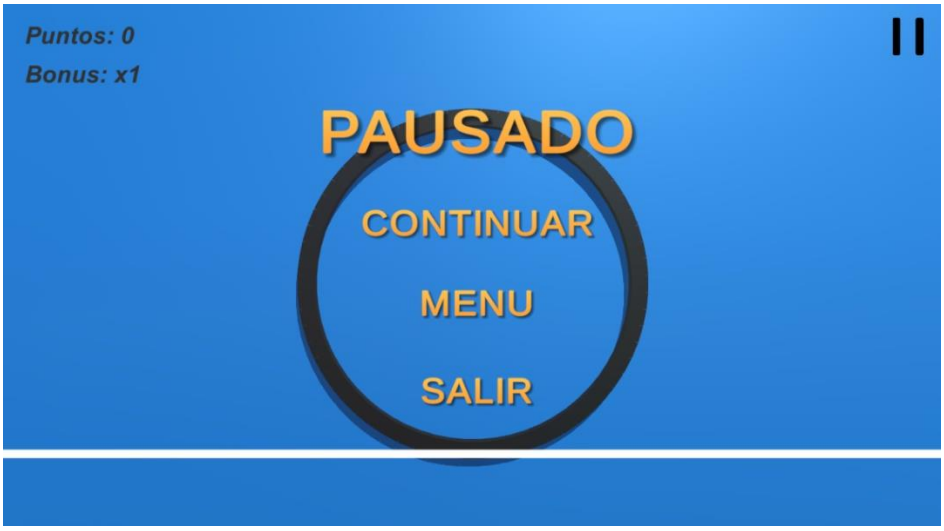


Ilustración 24 Menú de pausa (Marcos Sierra, 2019)

A3.1 Sprites y Efectos de Objetos

Los sprites y efectos de objetos son todos los diseños artísticos hechos para las notas a ser presionadas y los efectos que se muestran al presionarlas en distintos momentos. Estos diseños se crearon utilizando Photoshop CS6 y pueden ser apreciados en la Ilustración 25.

Ilustración 25 Sprites y Efectos de Objetos (Marcos Sierra, 2019)

A3.2 Diseño de Fondo de Nivel

El diseño de fondo de nivel fue desarrollado con la idea de que cada vez que se juega un nivel el color del fondo cambiará y se degrade a través de dos colores. En la Ilustración 26 se enseña el código utilizado para lograr este efecto, donde primero se genera un número aleatorio en la escala de color en los niveles de rojo, verde y azul, y se compara cual es el valor dominante para establecerlo como el primer color con una tonalidad profunda, y el segundo valor como una variación de éste para que el cambio que se produzca en la pantalla sea agradable a la vista.

```
byte R = (byte)Random.Range(0, 256);
byte G = (byte)Random.Range(0, 256);
byte B = (byte)Random.Range(0, 256);

if(R > G)
{
    if(G > B)
    {
        B = 0;
        G = 170;
        maxColor = new Color32(R, 0, 0, 0);
    }
    else if(B > R)
    {
        G = 0;
        R = 170;
        maxColor = new Color32(0, 0, B, 0);
    }
    else
    {
        G = 0;
        B = 170;
        maxColor = new Color32(R, 0, 0, 0);
    }
}
else
{
    if(R > B)
    {
        B = 0;
        R = 170;
        maxColor = new Color32(0, G, 0, 0);
    }
    else if(B > G)
    {
        R = 0;
        G = 170;
        maxColor = new Color32(0, 0, B, 0);
    }
    else
    {
        R = 0;
        G = 170;
        maxColor = new Color32(0, G, 0, 0);
    }
}
minColor = new Color32(R, G, B, 0);
```

Ilustración 26 Código para generar colores de fondo aleatorios (Marcos Sierra, 2019)

A3.3 Sprites y Efectos de Botones

Los sprites y efectos de botones se logran mediante otro recurso gratis de la tienda de recursos de Unity llamado TextMeshPro, el cual da varias opciones de edición de texto y efectos de sombreado al presionar el botón para dar una sensación más realista. Los sprites en este caso representan los mapas de textura utilizados para dar sombras y colores al texto de los botones.

A3.4 Diseño de Fondo de Menú

El diseño de fondo de menú consta de una imagen estática de fondo en los distintos menús para generar una visión más estética y evitar que los menús luzcan simples y robustos, además se combina los efectos de los botones y texto para obtener resultados como lo presentados en la Ilustración 27.

Ilustración 27 Diseño de Menús (Marcos Sierra, 2019)

- Fin Iteración 3 –

Finalizando las actividades planificadas para la Iteración 3 se termina la fase de Elaboración y se obtiene un juego funcional, con diseños agradables e intuitivos como se observa en la Ilustración 28 donde se mezclan los diseños de objetos y de fondo de nivel.

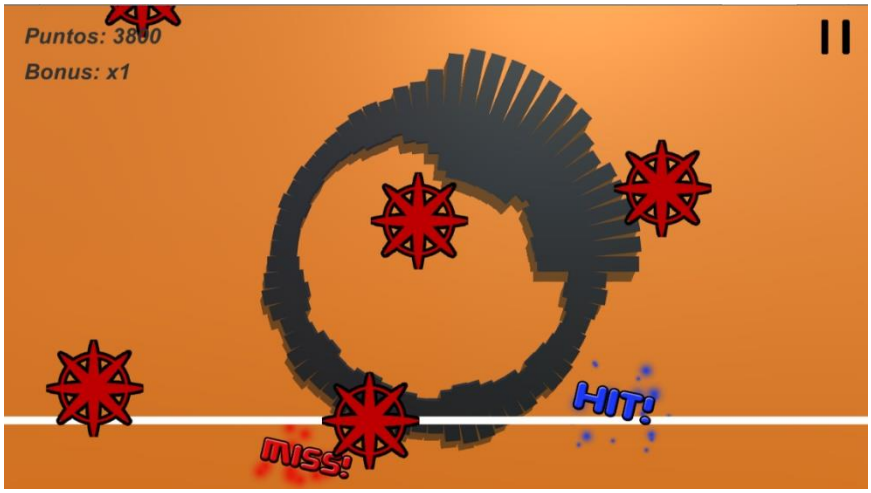


Ilustración 28 Juego Final (Marcos Sierra, 2019)

4.4 Fase de Pruebas

El propósito de la fase Beta es probar el funcionamiento del juego cuando se desea recibir la retroalimentación del Cliente. En el caso del presente proyecto hubo una Beta para cada Iteración, sin embargo esto no es el estándar necesariamente. Las Betas pueden realizarse cuando se considere necesario ya sea para comprobar ciertas funcionalidades críticas o analizar las opiniones del público acerca del producto. El formato utilizado para realizar las pruebas se encuentra en el Apéndice I, y los documentos de las pruebas realizadas se encuentran en los Apéndices J, K y L.

Las pruebas realizadas en este trabajo son de muy alto nivel donde el objetivo de ellas es analizar el comportamiento del software para asegurar el funcionamiento correcto del juego, sin embargo en entornos más profesionales es recomendable utilizar compañías que se especializan en la prueba de videojuegos, en caso de no contar con un departamento propio dedicado a pruebas.

También existen pruebas de funcionalidades donde se selecciona, ya sea al azar o conscientemente, a un grupo de personas para que prueben el juego con las nuevas implementaciones y aporten con su retroalimentación, a través de opiniones personales sobre la experiencia o errores que encontraron durante la ejecución, lo cual se obtiene mediante encuestas o entrevistas directas con el grupo.

En general no existen muchas herramientas para las pruebas de videojuegos ya que su enfoque es la interacción del jugador con la aplicación, por lo cual la práctica más común en la industria es contratar servicios de testers especializados en videojuegos los cuales después de un periodo de tiempo utilizando

la aplicación envían reportes de rendimiento, bugs, consumo de memoria, caída de cuadros por segundo, entre otros, y esta información es utilizada para ajustar los parámetros del juego y mejorar la calidad.

Sin embargo incluso estas tareas pueden ser automatizadas hasta cierto punto, una forma de hacerlo es utilizando bots para que repitan un tarea varias veces y analizar el comportamiento del software en respuesta a estas acciones. Una herramienta que se recomienda para estas acciones es "T-Plan Robot Enterprise", que puede ser utilizado para pruebas a nivel de GUI donde se escribe y ejecuta un script con los pasos a seguir para obtener un resultado, y el programa lo ejecuta similar a como lo haría una persona. Para pruebas de rendimiento en dispositivos móviles existe "GameBench" que provee estadísticas de cuadros por segundo, cuellos de botella, tiempos de carga, consumo de memoria y batería, y otras características útiles para optimizar la fluidez con la que se ejecuta el videojuego.

Volviendo al ejemplo, para la Iteración 1 se realizó una Beta con el propósito de hacer pruebas de funcionamiento, registro de datos y la precisión con la que puede interactuar el usuario. Se encontró que en el movimiento de los objetos no hubo mayores inconvenientes, simplemente era necesario ajustar un poco los valores para que el jugador pueda presionar los objetivos. En cuanto al registro de datos, éste funcionaba sin problemas, cada vez que se presionaba un objeto se obtenía el puntaje adecuado.

En la precisión con la que se presionaba los objetos se encontraban situaciones que al presionar demasiado temprano, se destruía el objeto pero no registraba ninguna puntuación porque no se encontraba en el rango aceptable de ningún puntaje. Para aquello se hizo una corrección rápida del Hit Box del objeto y se ajustaron los rangos de coordenadas de los puntajes para incluir este tipo de posibilidades.

En la Iteración 2 la Beta fue sin inconvenientes ya que solo era necesario probar los menús y submenús desarrollados y ya que su propósito es en mayor parte cambiar de escenas, no se generaron problemas. Pero al realizar las pruebas se encontraron posibles adiciones al juego que podrían volverlo más entretenido de jugar, estas son la inclusión de un menú de pausa y tener la opción de escoger la dificultad deseada, los cuales fueron añadidos en la siguiente fase de Planificación.

Finalmente en la Iteración 3 la última Beta realizada fue con el propósito de comprobar la correcta ejecución del juego una vez implementadas todas las funcionalidades, en la cual los únicos inconvenientes encontrados fueron que los íconos dentro de un dispositivo móvil eran demasiado grandes y que las dificultades eran más difíciles de lo esperado. Para solucionarlo se ajustó el tamaño de los objetos para que se vean mejor en la pantalla y se alteraron los valores de las distintas dificultades para que fuesen posibles de ser jugadas en el entorno real.

4.5 Fase de Cierre

Para la fase de Cierre se utiliza el Apéndice D, en donde se documenta dentro de un Postmortem las características finales del proyecto para contar con un registro que se puede consultar para futuros proyectos.

A continuación se presenta el documento generado durante la fase de Cierre detallando los aspectos correspondientes:

Postmortem

Duración del Proyecto El proyecto duró tres semanas iniciando el 09/09/2019 y finalizando el 30/09/2019.

Costo del Proyecto Ya que se utilizaron herramientas y recursos gratis no hubo ningún costo.

Problemas Encontrados

Se encontraron problemas de carga de audio al juego inicialmente debido a codificación errónea, falta de precisión en la interacción del usuario con el programa, diseños de objetos de tamaños demasiado grandes para las pantallas de dispositivos móviles, y valores de velocidad y cantidad de objetos generados que no permitían jugar el videojuego de forma satisfactoria.

Soluciones

Para la carga de audio se revisó la documentación de Unity y se buscó en foros posibles soluciones con lo cual se encontró el error de codificación y se corrigió sin mayor problema. En la falta de precisión se ajustaron los valores del Hit Box de los objetos y los rangos de aceptación de los puntajes.

Para los diseños solo fue necesario disminuir el tamaño para que fuesen mejor acoplados en pantallas móviles, y para los problemas de velocidad y cantidad de objetos, se probaron distintos valores hasta encontrar aquellos que se consideraban aceptables.

Funcionalidades Reusables

El funcionamiento base del juego puede ser reutilizado para distintos juegos rítmicos futuros y algunos de los diseños de los objetos pueden ser utilizados no solo en juegos rítmicos sino también en otros juegos donde la precisión es esencial para conseguir mejor puntuación.

Posibles Mejoras

Las mejoras que se pueden conseguir en el proyecto actual es la adición de un modo multijugador online, ya sea para jugar contra otros jugadores al mismo tiempo una canción y obtener la mejor puntuación, o jugar una canción en solitario y comparar la puntuación obtenida con otros jugadores. También, se podría considerar modos extra donde desaparecen momentáneamente los objetivos o el rango de visión es limitado.

Conclusión

En conclusión, el proyecto cumplió con los objetivos planteados al inicio del proceso, es decir se generó un videojuego móvil que cumple con las características definidas, el equipo logró aprender los aspectos relacionados con las distintas fases por las que pasa el desarrollo de un juego y los formatos que son necesarios de realizar en determinados puntos del proyecto, y por lo tanto se considera como un éxito.

CAPITULO 5: Conclusiones y recomendaciones

5.1 Conclusiones

- El trabajo desarrollado cumple con el objetivo principal planteado que es la elaboración de una guía metodológica para la creación de videojuegos vinculando metodologías de desarrollo. Se presenta un documento que explica las características que se consideran esenciales en el proceso de desarrollo de un videojuego, desde las metodologías que son utilizadas actualmente, los individuos que se encuentran en los equipos de trabajo y la documentación relacionada con cada etapa, hasta un ejemplo práctico usando la guía propuesta.
- En el trabajo se describen distintos motores de videojuegos con los que se puede crear aplicaciones de varias especificaciones y géneros diferentes, y se escogen las herramientas que mejor se acomodan con el ejemplo desarrollado. Sin embargo, se puede agregar, descartar o cambiar herramientas acorde al entorno de trabajo en el que se encuentre, ya que dependerá de lo que el equipo de trabajo y el cliente consideren como la mejor opción para determinado proyecto.
- Las metodologías de desarrollo usadas en la actualidad para la creación de videojuegos son las ágiles como Scrum y XP, pero aquellas metodologías no toman en cuenta roles particulares que se encuentran en la industria de videojuegos como son los equipos audiovisuales. La guía describe estos roles para una mejor comprensión del entorno en el que se trabaja detallando sus características y que funciones cumplen dentro del proceso de desarrollo.
- Se considera esencial el uso de documentación entregable en cada fase del proceso debido a que generan registros organizados que pueden ser utilizados en iteraciones futuras para mejorar el videojuego desarrollado, ya sea corrigiendo errores o añadiendo funcionalidades. Además, la documentación proporciona una base para la creación de nuevas aplicaciones mediante el reúso de características consideradas como éxitos de productos anteriores, agilizando la elaboración de proyectos similares.

- Gracias al uso de la guía propuesta en el ejemplo desarrollado se logró mantener un ambiente de trabajo organizado y ágil, con una clara imagen de las actividades a realizar durante cada iteración, tomando en cuenta el objetivo principal del proyecto y utilizando de forma eficaz la retroalimentación obtenida en la fase de pruebas para mejorar el videojuego.
- Una ventaja de la guía elaborada es su facilidad de implementación en diferentes proyectos, ya que no requiere de experiencia previa con otras metodologías y no tiene mayor complejidad. Además, es flexible porque la guía no se enfoca en un solo género de videojuego o plataforma, sino en el proceso mismo del desarrollo de la aplicación.
- Otro beneficio de la guía es que, debida a su ciclo de vida evolutivo, pueden utilizarse las fases de planificación, elaboración y beta, para implementar nuevas funcionalidades incluso después de haber sido terminado el videojuego, ya sea como parches o elementos totalmente nuevos.

5.2 Recomendaciones

- Evaluar distintas metodologías ágiles en un proyecto determinado y realizar una comparación de resultados con la guía propuesta.
- Utilizar la guía propuesta en un proyecto real con el propósito de analizar la eficiencia con la que se puede desarrollar un videojuego por un equipo de trabajo familiarizado con las herramientas en un entorno profesional.
- Se recomienda utilizar la guía en equipos de trabajo no mayores a diez personas, ya que el proceso de desarrollo de un videojuego se encuentra en constante cambio debido a los jugadores y sus demandas, por lo cual compartir las modificaciones en grupos de trabajo extensos resulta lento e incluso puede generar más errores de lo anticipado. Para equipos de trabajo mayores al número recomendado, el líder del equipo será el encargado de escoger la metodología que mejor se acomode a la situación.
- Se recomienda realizar pruebas periódicas del videojuego, de preferencia después de cada iteración, porque cambios que pueden parecer insignificantes tienen el potencial de afectar sustancialmente el comportamiento de objetos o variables dentro de la aplicación. Por ejemplo,

ajustes de tamaño de los objetos puede resultar en sobre posicionamiento de los mismos y causar movimientos inesperados.

- Es recomendable que las iteraciones planificadas no superen las dos semanas debido a que se requiere retroalimentación del cliente constantemente porque si se añaden funcionalidades grandes y no se encuentran de acuerdo a lo deseado por el cliente, modificarlas representaría un aumento considerable en el tiempo de entrega, y por consiguiente un aumento en el costo.
- Se recomienda utilizar la guía para familiarizarse con el proceso de creación de videojuegos, ya que permitirá adquirir conocimientos acerca de los pasos involucrados en el desarrollo de un videojuego, mientras se sigue un modelo ordenado y simple de entender.

Bibliografía

Acerenza, N., Coppes, A., Mesa, G., Viera, A., Fernández, E., Laurenzo, T., & Vallespir, D. (2009). *Una Metodología para Desarrollo de Videojuegos.* Montevideo: UR. FI – INCO.

Chromatic. (2003). *Extreme Programming Pocket Guide.* O'Reilly Media.

Elecbyte. (11 de Marzo de 2019). *Home*. Obtenido de M.U.G.E.N.: https://web.archive.org/web/20090925152230/http://www.elecbyte.com/mugen

Epic Games, Inc. (10 de Marzo de 2019). *About*. Obtenido de Unreal Engine: https://www.unrealengine.com/en-US/what-is-unreal-engine-4

Fernandez, D., & Angelina, C. (2011). *Desarrollo de Videojuegos: Arquitectura del Motor de Videojuegos.* Castilla: Bubok.

Juárez, A., & Mombiela, T. (2011). *Los videojuegos.* Barcelona: UOC.

Ren'Py. (14 de Marzo de 2019). *Ren'Py*. Obtenido de Ren'Py: https://www.renpy.org

RPG Maker. (14 de Marzo de 2019). *Home*. Obtenido de RPG Maker: https://www.rpgmakerweb.com/

Schwaber, K. (2004). *Agile Project Management with Scrum.* Washington: Microsoft.

Unity Technologies. (10 de Marzo de 2019). *Features*. Obtenido de Unity: https://unity3d.com/unity

YoYo Games. (10 de Marzo de 2019). *Features*. Obtenido de YoYo Games: https://www.yoyogames.com/gamemaker/features

Apéndice

Apéndice A. Formato de Entregable de la Fase de Concepto

Concepto

Proyecto: [Nombre]
Revisión 0.00

Fecha

Documento validado por las partes en fecha: [Fecha]

Por el cliente	Por la empresa suministradora
[Nombre]	[Nombre]

Fase Concepto

Modelo de Monetización

[Descripción de los sistemas de monetización integrados en el juego o medio por el cual se vende el producto. Por ejemplo: micro transacciones por Google Play, Apple Store, venta a través de la plataforma Steam, publicidad dentro del juego, etc…]

Características del Videojuego

Categorías	**Detalle**
Narrativa	Si el juego cuenta con una historia o no
Cantidad de Jugadores	Número de jugadores que se encuentran conectados en el mismo dispositivo o partida
Online	Si el juego requiere de internet para ser utilizado. Ejemplo: Si es principalmente multijugador o contiene elementos que se acceden a través de una conexión a internet
Cooperativo o Competitivo	La forma en la que interactúan los jugadores
Género(s)	El tipo de juego que se desea
Temática	Entorno en el que se desenvuelve el jugador. Por ejemplo: medieval, futurista, fantasía, etc…

Descripción General

[Se describe la idea general del juego, es decir, la manera en la que se juega, el propósito del juego, condiciones de victoria u otros aspectos que se consideren necesarios]

Apéndice B. Formato de Entregable de la Fase de Planificación

Planificación

Proyecto: [Nombre]
Revisión 0.00

Fecha

Documento validado por las partes en fecha: [Fecha]

Por el cliente	Por la empresa suministradora
[Nombre]	[Nombre]

Fase de Planificación

Objetivos Generales

[Se definen los objetivos y resultados que se esperan obtener al finalizar el proyecto]

Objetivos Específicos de la Iteración

[Se definen los objetivos que se desean alcanzar al finalizar la iteración]

Cronograma de Trabajo

Funcionalidad	Actividades	Área Responsable	Requisitos Previos	Prioridad	Fecha Inicio	Fecha Fin Planeada	Fecha Fin Real
F1. Funcionalidad 1	A1.1 Actividad 1.1	Desarrollo		Alta	XXXX-XX-XX	XXXX-XX-XX	XXXX-XX-XX
	A1.2 Actividad 1.2	Diseño		Baja	XXXX-XX-XX	XXXX-XX-XX	XXXX-XX-XX
	A1.3 Actividad 1.3	Sonido	A1.2	Media	XXXX-XX-XX	XXXX-XX-XX	XXXX-XX-XX
F2. Funcionalidad 2	A2.1 Actividad 2.1	Desarrollo		Alta	XXXX-XX-XX	XXXX-XX-XX	XXXX-XX-XX
	A2.2 Actividad 2.2	Diseño	F1	Alta	XXXX-XX-XX	XXXX-XX-XX	XXXX-XX-XX
F3. Funcionalidad 3	A3.1 Actividad 3.1	Desarrollo	F2	Media	XXXX-XX-XX	XXXX-XX-XX	XXXX-XX-XX
	A3.2 Actividad 3.2	Sonido		Baja	XXXX-XX-XX	XXXX-XX-XX	XXXX-XX-XX
	A3.3 Actividad 3.3	Diseño	F1, F2	Baja	XXXX-XX-XX	XXXX-XX-XX	XXXX-XX-XX
	A3.4 Actividad 3.4	Desarrollo	A3.1, A3.2	Alta	XXXX-XX-XX	XXXX-XX-XX	XXXX-XX-XX

Apéndice C. Formato de Especificación de Requerimientos

Especificación de requerimientos

Proyecto: [Nombre]
Revisión 0.00

Fecha

Documento validado por las partes en fecha: [Fecha]

Por el cliente	Por la empresa suministradora
[Nombre]	[Nombre]

1 Descripción general

1.1 Funcionalidades del videojuego

[Se identifican las funcionalidades con las que cuenta el juego, como ejemplo: puntuaciones, rankings, creación de personaje personalizado, personalización de la interfaz gráfica, mensajes internos, entre otros.]

1.2 Características de los usuarios

[Se especifica el público objetivo del juego.]

1.3 Requisitos técnicos

[Se describen las especificaciones técnicas del hardware en donde se ejecutará el juego, las herramientas a utilizar en el desarrollo, la plataforma, el sistema operativo, librerías y dependencias]

1.4 Crecimiento esperado

[Se estima el aumento de funcionalidades y contenido que puede presentarse a futuro]

2 Requerimientos específicos

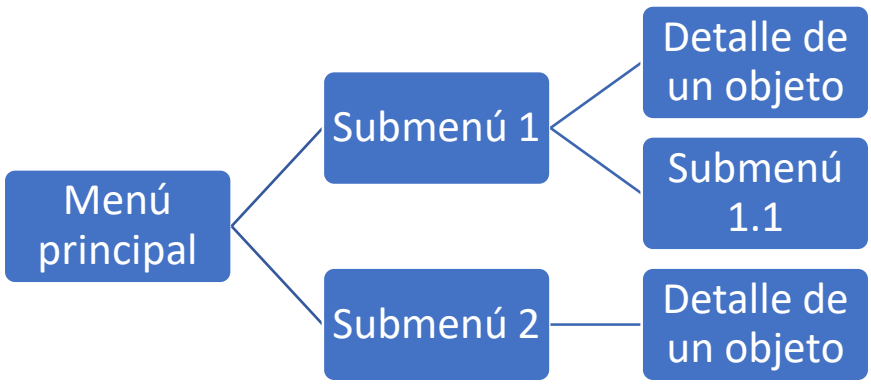
Número de funcionalidad			
Nombre de funcionalidad			
Tipo	**X** Sistema	☐ Contenido	
Prioridad del requisito	**X** Alta/Esencial	☐ Media/Deseado	☐ Baja/ Opcional

2.1 Prototipos

[Se colocan imágenes iniciales de las funcionalidades a implementar]

2.2 Flujo de Interfaz

[Flujograma de la jerarquía de la interfaz]



2.3 Requerimientos no funcionales

2.3.1 Requisitos de rendimiento

[Velocidad de carga para iniciar el juego, tiempo de cambio entre pantallas, tiempo de respuesta con el servidor…]

2.3.2 Seguridad

[Métodos de inicio de sesión, encriptación de los datos del usuario o sistema, medidas anti hacker…]

2.3.3 Fiabilidad

[Tolerancia permitida frente a posibles fallos, es decir cuales datos se pueden perder sin mayor consecuencia o cual es el tiempo máximo de fallo del sistema que se considera aceptable]

2.3.4 Disponibilidad

[Se establece un tiempo estándar dentro del cual se garantiza que el juego se mantiene en funcionamiento, incluso si se presentan fallos en él]

2.3.5 Mantenibilidad

[La mantenibilidad se refiere a que tan preparado está el sistema para mejoras de funcionamiento, corrección de fallos o adaptación a un nuevo entorno. Además, se define los periodos de mantenimiento y los responsables del mismo]

2.3.6 Portabilidad

[La portabilidad es la facilidad que tiene un sistema para ser trasladado y usado en otra plataforma o hardware. Considera el lenguaje en el que fue programado, el sistema operativo, la computadora, entre otras variables]

Apéndice D. Formato del Postmortem

Postmortem

Proyecto: [Nombre]
Revisión 0.00

Fecha

Documento validado por las partes en fecha: [Fecha]

Por el cliente	Por la empresa suministradora
[Nombre]	[Nombre]

Postmortem

Duración del Proyecto [Tiempo total de Desarrollo]

Costo del Proyecto [Costo total]

Problemas Encontrados

[Inconvenientes de tiempo, errores del sistema, falta de presupuesto…]

Soluciones

[Formas en las que se solucionó los problemas]

Funcionalidades Reusables

[Aspectos que pueden ser usados en otros juegos, como base o en su totalidad]

Posibles Mejoras

[Recomendaciones para proyectos futuros o aumentos en las funcionalidades del proyecto actual]

Conclusión

[Opinión final del proceso de trabajo, la interacción y productividad del equipo, y el éxito o fracaso del proyecto]

Apéndice E. Especificación de Requerimientos de la Iteración 1

Especificación de requerimientos

Proyecto: Videojuego Rítmico con Música Personal
Revisión 1.00

09/09/2019

Documento validado por las partes en fecha: 09/09/2019

Por el cliente	Por la empresa suministradora
Marcos Sierra	Arachnyx

1 Descripción general

1.1 Funcionalidades del videojuego

El videojuego permite al usuario presionar objetos en el momento adecuado para conseguir puntos en relación al tiempo y lugar que fue oprimido el objeto en movimiento.
Cuenta con un sistema de puntuación, un menú principal básico, un menú de puntuación final y la funcionalidad de escoger música personal para jugar.

1.2 Características de los usuarios

El público al que se apunta son los amantes de juegos rítmicos y juegos en los que la precisión representa un gran papel para conseguir una mejor puntuación. Además, jugadores casuales que quieren relajarse durante el día o pasar el tiempo mientras están esperando algún evento.

1.3 Requisitos técnicos

El sistema operativo en el que se ejecutará el juego es en dispositivos móviles con sistema operativo Android de versiones 9.0 y mayor, ya que está fue la versión que fue lanzada a finales del 2018, por lo cual podrá tener una larga vida útil. Se desarrollará el producto utilizando el motor de videojuegos Unity 3D versión 2019.2.4f1, usando el SDK de Android que puede ser instalado junto con el motor de videojuegos.
Para diseño artístico se utiliza Adobe Photoshop CS6. Debido a la naturaleza del videojuego no es necesario diseño de audio y por lo tanto no se utilizarán herramientas relacionadas a ello.

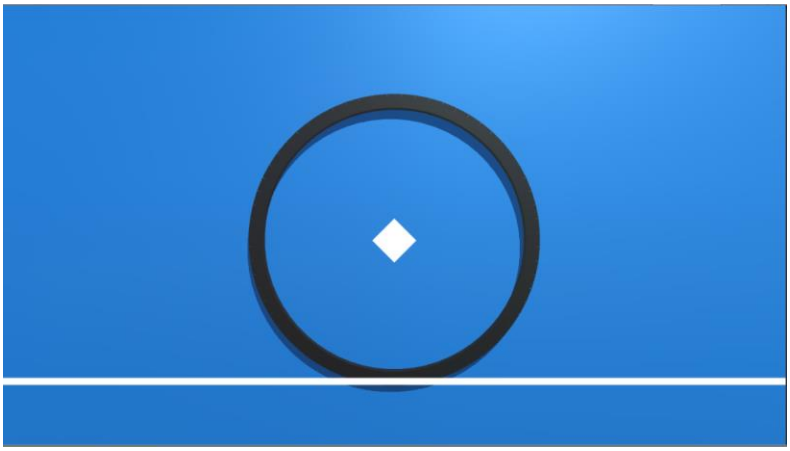
1.4 Crecimiento esperado

A futuro se plantea implementar un sistema de juego online y competitivo mediante canciones predefinidas incluidas en el videojuego con las cuales se puede clasificar a los jugadores.

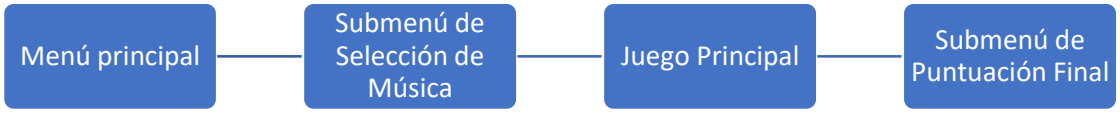
2 Requerimientos específicos

Número de funcionalidad	F1		
Nombre de funcionalidad	Base del Juego		
Tipo	**X** Sistema	☐ Contenido	
Prioridad del requisito	**X** Alta/Esencial	☐ Media/Deseado	☐ Baja/ Opcional

2.1 Prototipos



2.2 Flujo de Interfaz



2.3 Requerimientos no funcionales

2.3.1 Requisitos de rendimiento

La velocidad de carga del juego no debe ser mayor a 2 segundos y la velocidad de cambio de pantallas debe ser máximo 1 segundo.

2.3.2 Seguridad

La seguridad es mínima debido a que no se requiere de los datos del usuario para jugarlo y no se guarda información personal.

2.3.3 Fiabilidad

El tiempo máximo de fallo permitido es de 3 segundos, después de pasar esta duración es necesario reiniciar el juego, salir de él o mostrar un mensaje de error.

2.3.4 Disponibilidad

Se garantiza que el juego se mantiene en funcionamiento sin necesidad de conexión a internet. El único caso en el que el juego no puede ser utilizado es cuando el usuario no tiene música propia dentro del dispositivo.

2.3.5 Mantenibilidad

Debido a la simpleza del sistema, este siempre está disponible a mejoras y aumento de funcionalidades sin necesidad de cambios mayores.

2.3.6 Portabilidad

Ya que el juego es desarrollado en Unity 3D es posible el cambio de plataforma sin grandes problemas por sus funciones, los únicos cambios que se deberían considerar son la interacción del usuario con el juego dependiendo de los periféricos de entrada con los que cuenta la nueva plataforma objetivo.

Apéndice F. Especificación de Requerimientos de la Iteración 2

Especificación de requerimientos

Proyecto: Videojuego Rítmico con Música Personal
Revisión 2.00

16/09/2019

Documento validado por las partes en fecha: 16/09/2019

Por el cliente	Por la empresa suministradora
Marcos Sierra	Arachnyx

1 Descripción general

1.1 Funcionalidades del videojuego

El videojuego permite al usuario presionar objetos en el momento adecuado para conseguir puntos en relación al tiempo y lugar que fue oprimido el objeto en movimiento.
Cuenta con un sistema de puntuación, un menú principal básico, un menú de puntuación final y la funcionalidad de escoger música personal para jugar.

1.2 Características de los usuarios

El público al que se apunta son los amantes de juegos rítmicos y juegos en los que la precisión representa un gran papel para conseguir una mejor puntuación. Además, jugadores casuales que quieren relajarse durante el día o pasar el tiempo mientras están esperando algún evento.

1.3 Requisitos técnicos

El sistema operativo en el que se ejecutará el juego es en dispositivos móviles con sistema operativo Android de versiones 9.0 y mayor, ya que está fue la versión que fue lanzada a finales del 2018, por lo cual podrá tener una larga vida útil. Se desarrollará el producto utilizando el motor de videojuegos Unity 3D versión 2019.2.4f1, usando el SDK de Android que puede ser instalado junto con el motor de videojuegos.
Para diseño artístico se utiliza Adobe Photoshop CS6. Debido a la naturaleza del videojuego no es necesario diseño de audio y por lo tanto no se utilizarán herramientas relacionadas a ello.

1.4 Crecimiento esperado

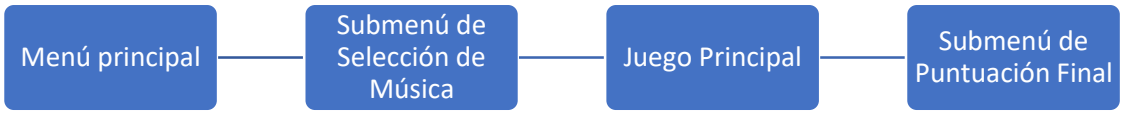
A futuro se plantea implementar un sistema de juego online y competitivo mediante canciones predefinidas incluidas en el videojuego con las cuales se puede clasificar a los jugadores.

2 Requerimientos específicos

Número de funcionalidad	F2		
Nombre de funcionalidad	Menús		
Tipo	**X** Sistema	☐ Contenido	
Prioridad del requisito	**X** Alta/Esencial	☐ Media/Deseado	☐ Baja/ Opcional

2.1 Prototipos

2.2 Flujo de Interfaz



2.3 Requerimientos no funcionales

2.3.1 Requisitos de rendimiento

La velocidad de carga del juego no debe ser mayor a 2 segundos y la velocidad de cambio de pantallas debe ser máximo 1 segundo.

2.3.2 Seguridad

La seguridad es mínima debido a que no se requiere de los datos del usuario para jugarlo y no se guarda información personal.

2.3.3 Fiabilidad

El tiempo máximo de fallo permitido es de 3 segundos, después de pasar esta duración es necesario reiniciar el juego, salir de él o mostrar un mensaje de error.

2.3.4 Disponibilidad

Se garantiza que el juego se mantiene en funcionamiento sin necesidad de conexión a internet. El único caso en el que el juego no puede ser utilizado es cuando el usuario no tiene música propia dentro del dispositivo.

2.3.5 Mantenibilidad

Debido a la simpleza del sistema, este siempre está disponible a mejoras y aumento de funcionalidades sin necesidad de cambios mayores.

2.3.6 Portabilidad

Ya que el juego es desarrollado en Unity 3D es posible el cambio de plataforma sin grandes problemas por sus funciones, los únicos cambios que se deberían considerar son la interacción del usuario con el juego dependiendo de los periféricos de entrada con los que cuenta la nueva plataforma objetivo.

Apéndice G. Especificación de Requerimientos de la Iteración 3

Especificación de requerimientos

Proyecto: Videojuego Rítmico con Música Personal
Revisión 3.00

20/09/2019

Documento validado por las partes en fecha: 20/09/2019

Por el cliente	Por la empresa suministradora
Marcos Sierra	Arachnyx

1 Descripción general

1.1 Funcionalidades del videojuego

El videojuego permite al usuario presionar objetos en el momento adecuado para conseguir puntos en relación al tiempo y lugar que fue oprimido el objeto en movimiento.
Cuenta con un sistema de puntuación, un menú principal básico, un menú de puntuación final y la funcionalidad de escoger música personal para jugar.

1.2 Características de los usuarios

El público al que se apunta son los amantes de juegos rítmicos y juegos en los que la precisión representa un gran papel para conseguir una mejor puntuación. Además, jugadores casuales que quieren relajarse durante el día o pasar el tiempo mientras están esperando algún evento.

1.3 Requisitos técnicos

El sistema operativo en el que se ejecutará el juego es en dispositivos móviles con sistema operativo Android de versiones 9.0 y mayor, ya que está fue la versión que fue lanzada a finales del 2018, por lo cual podrá tener una larga vida útil. Se desarrollará el producto utilizando el motor de videojuegos Unity 3D versión 2019.2.4f1, usando el SDK de Android que puede ser instalado junto con el motor de videojuegos.
Para diseño artístico se utiliza Adobe Photoshop CS6. Debido a la naturaleza del videojuego no es necesario diseño de audio y por lo tanto no se utilizarán herramientas relacionadas a ello.

1.4 Crecimiento esperado

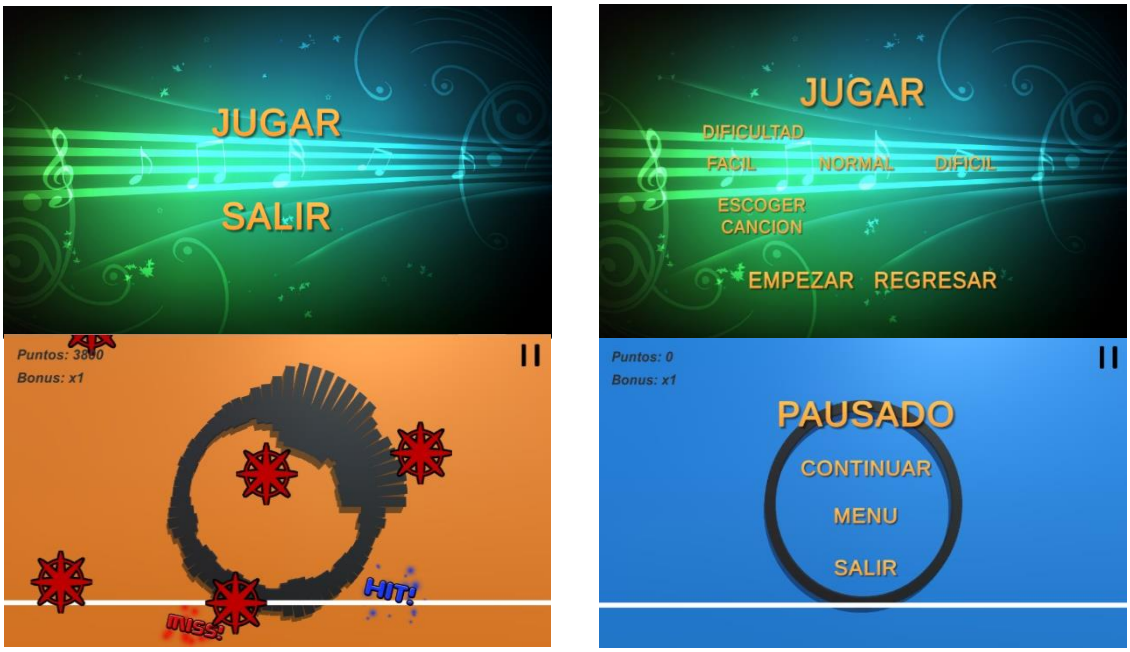
A futuro se plantea implementar un sistema de juego online y competitivo mediante canciones predefinidas incluidas en el videojuego con las cuales se puede clasificar a los jugadores.

2 Requerimientos específicos

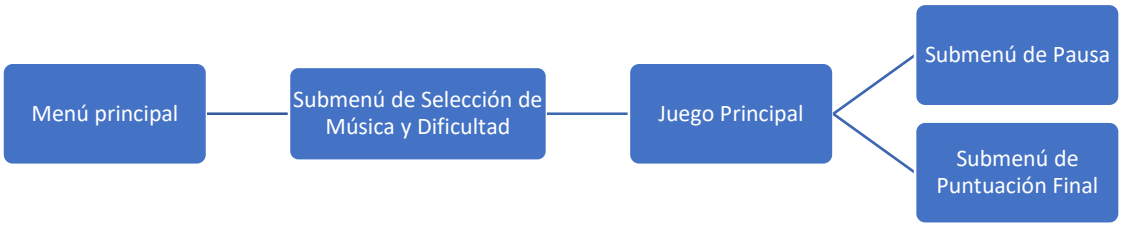
Número de funcionalidad	F2.A2.4		
Nombre de funcionalidad	Menú de Pausa y Opción de Dificultad		
Tipo	**X** Sistema	☐ Contenido	
Prioridad del requisito	**X** Alta/Esencial	☐ Media/Deseado	☐ Baja/ Opcional

Número de funcionalidad	F3		
Nombre de funcionalidad	Diseño Artistico		
Tipo	**X** Sistema	☐ Contenido	
Prioridad del requisito	**X** Alta/Esencial	☐ Media/Deseado	☐ Baja/ Opcional

2.1 Prototipos



2.2 Flujo de Interfaz



2.3 Requerimientos no funcionales

2.3.1 Requisitos de rendimiento

La velocidad de carga del juego no debe ser mayor a 2 segundos y la velocidad de cambio de pantallas debe ser máximo 1 segundo.

2.3.2 Seguridad

La seguridad es mínima debido a que no se requiere de los datos del usuario para jugarlo y no se guarda información personal.

2.3.3 Fiabilidad

El tiempo máximo de fallo permitido es de 3 segundos, después de pasar esta duración es necesario reiniciar el juego, salir de él o mostrar un mensaje de error.

2.3.4 Disponibilidad

Se garantiza que el juego se mantiene en funcionamiento sin necesidad de conexión a internet. El único caso en el que el juego no puede ser utilizado es cuando el usuario no tiene música propia dentro del dispositivo.

2.3.5 Mantenibilidad

Debido a la simpleza del sistema, este siempre está disponible a mejoras y aumento de funcionalidades sin necesidad de cambios mayores.

2.3.6 Portabilidad

Ya que el juego es desarrollado en Unity 3D es posible el cambio de plataforma sin grandes problemas por sus funciones, los únicos cambios que se deberían considerar son la interacción del usuario con el juego dependiendo de los periféricos de entrada con los que cuenta la nueva plataforma objetivo.

Apéndice H. Estándares de Desarrollo

Estándares de Desarrollo

Proyecto: Videojuego Rítmico con Música Personal
Revisión 1.00

09/09/2019

Documento validado por las partes en fecha: 09/09/2019

Por el cliente	Por la empresa suministradora
Marcos Sierra	Arachnyx

1. Comentarios

Los comentarios se dividen en tres tipos: los comentarios de bloque, los comentarios de línea y los comentarios de final de línea.

- Comentarios de Bloque

  Los comentarios de bloque se utilizan cuando se requiere describir claramente la funcionalidad de un grupo de código, estructura de dato o algoritmo, y se necesita de varias líneas para dejar claro el funcionamiento.

  Ejemplo:

  ```
  /*
  Comentario
  De
  Bloque
  …
  */
  ```

- Comentarios de Línea

  Los comentarios de línea se utilizan para describir las líneas de código que les siguen y ocupan solamente una línea y deben tener un espacio en blanco encima suyo.

  Ejemplo:

  ```
  // Comentario de Línea…
  ```

- Comentarios de Final de Línea

  Los comentarios de final de línea se colocan después de una línea de código y son usados para detallar aspectos claves o específicos que se encuentran en el código al que están adjuntos.

  Ejemplo:

  ```
  string  rangoValor = “F”; // Valor al obtener menos del 40% de aciertos
  ```

2. Ubicación de Variables

Todas las variables que se utilizan en más de un método deben colocarse al inicio del Script.
En caso de que las variables sean usadas en un solo método deben ser ubicadas al inicio del método.
Para las variables que se definen dentro de bucles, también deben ser escritas al inicio del bucle.

Ejemplo:

```
public void Metodo( )
{
    int limite = 5;
    bool condicion;

    …
}
```

3. Declaración de métodos

Para declarar métodos se sigue el siguiente formato:

- El nombre de todo método debe empezar con la primera letra en mayúscula y en caso de tener dos o más palabras, cada una debe empezar con mayúscula.
- El carácter de inicio de bloque "{", será escrito después de un salto de línea.
- El carácter de fin de bloque "}", será escrito en una línea sola sin sentencias presentes extra.
- Cada método debe estar separada por una línea en blanco.
- Los métodos siempre serán privados a excepción de métodos que deben ser accedidos desde otros Scripts.

Ejemplo:

```
public void EmpezarJuego()
{
    Código del
    Método
    …
}

private void NotaBuena()
{
    Código del
    Método
    …
}
```

4. Sentencias y Espacios en blanco

Las sentencias deben estar tabuladas hacia la derecha con respecto a la sentencia que las contiene y los espacios en blanco se utilizan para dar una mayor legibilidad del código y se aplica para:

- Separar las declaraciones de variables del resto del código. Ejemplo:

```
int variable1 = 3;
int variable2 = 2;
int variable3;

variable3 = variable1 + variable2;
```

- Para separar valor por comas. Ejemplo:

```
variable.MetodoAsociado(x, y, z);
```

- Para separar operadores en operaciones matemáticas. Ejemplo:

```
var1 = (var2 + var3) * (var2 / var3);
```

5. Declaración de variables

Las variables siempre empezarán con palabras en minúsculas y en caso de ser compuestas la primera palabra será escrita en minúsculas y las siguiente palabras empezarán por letras mayúsculas. Los nombres de las variables tienen que ser los más cortos posibles y describir claramente su uso o relación en el código. Esta regla no se aplica solamente a variables temporales.

Ejemplo:

```
AudioSource música;
string txtPuntajeTotal;
int tiempoEspera;
```

6. Paréntesis

Para sentencias donde se realicen varias operaciones o comparaciones, es recomendable utilizar paréntesis para agrupar y separar código que pueda resultar confuso o complicado de leer en el futuro.
Ejemplo:

```
a = b + c / d * b; // Esto no es aceptable

a = ((b + c) / d) * b; // Esto es correcto


while(x == y || y != z) //Incorrecto

while((x == y) || (y != z)) //Correcto
```

7. Scripts

Cada Script debe incluir solamente el funcionamiento relacionado con el objeto al que está asignado determinado Script, es decir, el Script que contiene el funcionamiento del objeto Planta no debe contener también el funcionamiento del objeto Animal.

8. Organización de archivos

Cada archivo debe estar ubicado en su carpeta correspondiente y no mezclarse con otros de distinto tipo. Por ejemplo, los Scripts y todo aquello relacionado al código debe estar en la carpeta de Scripts, los materiales de los objetos en la carpeta de Materiales, la música y pistas de audio en la carpeta Audio, los Sprites, imágenes y arte conceptual en la carpeta Imágenes, etc.

Apéndice I. Formato de Entregable de la Fase de Pruebas

Pruebas de Funcionamiento

Proyecto: [Nombre]
Revisión 0.00

Fecha

Documento validado por las partes en fecha: [Fecha]

Por el cliente	Por la empresa suministradora
[Nombre]	[Nombre]

Prueba X

[Nombre de la Prueba]
Descripción: [Descripción general de lo que se va a probar]
Pasos: [Los pasos a seguir para lograr el resultado esperado]
Resultado esperado: [Descripción del resultado que se espera obtener]
Resultado obtenido: [Descripción del resultado que se obtuvo al ejecutar los pasos]

Observaciones:

[Se anota posibles mejoras, los errores encontrados, modificaciones menores o lo que se considere necesario documentar para utilizarlo de retroalimentación para el siguiente ciclo]

Apéndice J. Pruebas de Funcionamiento de la Iteración 1

Pruebas de Funcionamiento

Proyecto: Videojuego Rítmico con Música Personal
Revisión 1.00

13/09/2019

Documento validado por las partes en fecha: 13/09/2019

Por el cliente	Por la empresa suministradora
Marcos Sierra	Arachnyx

Prueba 1

Movimiento de Objetos
Descripción: Confirmar que los objetos se mueven a una velocidad adecuada y en la dirección correcta.
Pasos: Iniciar el juego. Dejar que los objetos se muevan sin presionarlos.
Resultado esperado: Los objetos deben descender a un ritmo constante y a una velocidad que el usuario pueda seguirlos sin mayor dificultad.
Resultado obtenido: Los objetos se mueven en la dirección esperada y de forma constante sin detenerse, sin embargo la velocidad es más rápida de lo aceptable por lo que es necesario ajustar los valores.

Prueba 2

Registro de Datos
Descripción: Confirmar que cuando se presiona un objeto, se registre la puntuación correspondiente.
Pasos: Iniciar el juego. Esperar a que los objetos se muestren por pantalla. Presionar los objetos en el momento indicado.
Resultado esperado: Al presionar los objetos el juego debe actualizar la puntuación aumentado el valor obtenido.
Resultado obtenido: El puntaje es actualizado correctamente y no se encontraron problemas.

Prueba 3

Precisión de Interacción del Usuario
Descripción: Confirmar que el juego detecte correctamente el Input del Usuario al presionar la pantalla.
Pasos: Iniciar el juego. Esperar a que los objetos se muestren por pantalla. Presionar los objetos en el momento indicado.
Resultado esperado: Al presionar los objetos estos deben ser destruidos y actualizar la puntuación.
Resultado obtenido: Los objetos son destruidos correctamente, sin embargo el Hit Box del objeto requiere de modificaciones ya que se presentan casos donde se destruye el objeto pero no se registra ninguna puntuación.

Observaciones:

Corregir los errores encontrados en el Movimiento de Objetos y el Registro de Datos.

Apéndice K. Pruebas de Funcionamiento de la Iteración 2

Pruebas de Funcionamiento

Proyecto: Videojuego Rítmico con Música Personal
Revisión 2.00

19/09/2019

Documento validado por las partes en fecha: 19/09/2019

Por el cliente	Por la empresa suministradora
Marcos Sierra	Arachnyx

Prueba 1

Menú Principal
Descripción: Confirmar el funcionamiento correcto de los botones del menú.
Pasos: Iniciar el juego. Presionar los distintos botones disponibles.
Resultado esperado: El botón de Iniciar debe cambiar de escena al submenú de selección de música y el botón Salir debe cerrar el juego.
Resultado obtenido: Los botones cumplen con el funcionamiento esperado.

Prueba 2

Submenú de Selección de Música
Descripción: Confirmar el funcionamiento correcto de la selección de música y el botón de Empezar a Jugar.
Pasos: Iniciar el juego. Presionar el botón Iniciar. Presionar el botón Seleccionar Música. Presionar el botón Empezar a Jugar.
Resultado esperado: Seleccionar Música debe permitir cargar la canción deseada y Empezar a Jugar debe cambiar de escena al juego principal.
Resultado obtenido: La música se puede seleccionar satisfactoriamente y el juego inicia al presionar Empezar a Jugar sin problemas.

Observaciones:

No se encontraron errores, pero al realizar las pruebas se encontró una posible adicción al juego para que sea más entretenido mediante el aumento de una opción para elegir la dificultad preferida y un menú de pausa en el juego principal.

Apéndice L. Pruebas de Funcionamiento de la Iteración 3

Pruebas de Funcionamiento

Proyecto: Videojuego Rítmico con Música Personal
Revisión 3.00

30/09/2019

Documento validado por las partes en fecha: 30/09/2019

Por el cliente	Por la empresa suministradora
Marcos Sierra	Arachnyx

Prueba 1

Ejecución Completa del Juego
Descripción: Confirmar el funcionamiento correcto de todo el juego en un dispositivo móvil.
Pasos: Iniciar el juego. Presionar los distintos botones disponibles. Seleccionar la música deseada. Escoger las distintas dificultades. Jugar el juego. Probar el menú de pausa. Visualizar la puntuación final.
Resultado esperado: Iniciar el juego después de seleccionar la música y dificultad deseada, jugar a través de la canción y que el juego muestre por pantalla el resultado obtenido.
Resultado obtenido: El juego se ejecuta de la forma esperada y la canción y dificultad cambia conforme a lo seleccionado, y el resultado mostrado al final corresponde a la puntuación obtenida durante el transcurso del nivel.

Observaciones:

No se encontraron errores de funcionalidad, sin embargo algunas dificultades requieren de ajustes para que se puedan jugar y los iconos en general deben ser modificados de tamaño ligeramente para que se vean bien en relación a la pantalla.