UNIVERSIDAD PÚBLICA DE EL ALTO

CARRERA INGENIERÍA DE SISTEMAS

PROYECTO DE GRADO

APLICACIÓN PARA LA MOTIVACIÓN DEL APRENDIZAJE DE LA MATEMÁTICA PARA SEXTO DE SECUNDARIA, UTILIZANDO GAMIFICACIÓN.

CASO: CENTRO DE MULTISERVICIOS EDUCATIVOS

Para optar el título de Licenciatura en Ingeniería de Sistemas

Mención: GESTIÓN Y PRODUCCIÓN

Postulante: Felix Benito Mamani Quispe

Tutor Metodológico: Ing. Marisol Arguedas Balladares

Tutor revisor: Ing. M.Sc. Gregorio Mamani Mamani

Tutor Especialista: Ing. Roberto David Mayta Aliaga

EL ALTO - BOLIVIA

2020

DEDICATORIA

Dedico este proyecto de grado a Dios, quien en su infinita misericordia me ha permitido llegar hasta este momento tan importante en mi formación profesional. A mi padres ya que me motivan a continuar día a día sin desfallecer, convertirme en su mejor ejemplo, ha forjado en mí un espíritu de fortaleza y perseverancia para cumplir mis objetivos y metas.

AGRADECIMIENTOS

Le doy gracias a mi Dios por permitirme vivir esta alegría, disfrutar cada día y serla persona que soy. A mis padre Benito, por darme la vida y por ser ese apoyo incondicional, gracias por estar pendientes de mí en cada momento, y enseñarme a vivir con responsabilidad, dedicación y esfuerzo, mil gracias por la hermosa familia que me han dado. A mis hermanos Beatriz, Carlos y Lucy, hemos crecido juntos y aprendimos a vivir, como amigos incondicionales compartiendo todo y aprendiendo que nuestras diferencias se convierten en riqueza cuando existe amistad y respeto, gracias por ser buenos hermanos.

Contenido

1. CAPITULO I: MARCO PRELIMINAR 1

1.1. Introducción 1

1.2. Antecedentes 1

1.2.1. Antecedentes institucionales 1

1.2.2. Antecedente internacional 3

1.2.3. Antecedente nacional 3

1.2.4. Antecedente local 4

1.3. Planteamiento del problema 4

1.3.1. Problema principal 5

1.3.2. Problema secundarios 5

1.4. Objetivos 5

1.4.1. Objetivo general 5

1.4.2. Objetivos específicos 5

1.5. Justificación 6

1.5.1. Justificación técnica 6

1.5.2. Justificación económica 6

1.5.3. Justificación social 6

1.6. Metodologías 7

1.7. Herramientas 7

1.8. Límites y alcances 7

1.8.1. Límites 7

1.8.2. Alcances 8

1.9. Aportes....8
2. CAPITULO II: MARCO TEÓRICO....9
2.1. Introducción....9
2.1.1. Conceptos....9
2.1.2. Dinámicas....9
2.1.3. Mecánicas....10
2.1.4. Componentes....10
2.2. Problemas en el Aprendizaje de la Matemática....11
2.2.1. Estilos de Aprendizaje....12
2.2.2. Pensamiento concentrado o difuso....13
2.2.3. Motivación....14
2.3. Framework D6 para el Diseño de Gamificación....15
2.3.1. Definir los Objetivos de Negocio:....16
2.3.2. Delinear los Comportamientos Objetivo....16
2.3.3. Describir a los Jugadores....16
2.3.4. Determinar los Ciclos de Actividad....16
2.3.5. ¿Dónde se Encuentra la Diversión?....16
2.3.6. Desplegar las Herramientas Adecuadas....17
2.4. Metodología SUM....17
2.4.1. Etapas....18
2.4.2. Ventajas y Desventajas....20
2.5. Costo y/o beneficio....21
2.5.1. Método de COCOMO....21
2.6. METRICAS DE CALIDAD....22
2.6.1. ISO/IEC 9126....22

2.7. Godot Engine....30
2.7.1. Escenas y nodos....31
2.7.2. Instanciar....32
2.7.3. Scripting....32
2.8. Arduino....33
2.8.1. Características....34
2.8.2. Programación....34
2.9. Realidad virtual....35
2.9.1. Tipos....36
2.9.2. Características....37
2.10. Sensores....39
2.10.1. MPU-6050....39
CAPITULO III: MARCO APLICATIVO....40
3.1. Ingeniería de requisitos....40
3.1.1. Requisitos funcionales....40
3.1.2. Requisitos no funcionales....40
3.2. Aplicación de la Metodología SUM....40
3.2.1. Fase 1: Concepto....40
3.2.2. Fase 2: Planificación....40
3.3. Diagrama de Actividades....43
3.2.2. Fase 3: Elaboración....44
3.2.3. Fase 4: Beta....48
3.2.4. Fase 5: Cierre....48
3.2. Aplicación de Framework D6....49
3.2.1. Definir los Objetivos de Negocio:....49

3.2.2. Delinear los Comportamientos Objetivo..... 49
3.2.3. Describir a los Jugadores..... 49
3.2.4. Determinar los Ciclos de Actividad..... 49
3.2.5. ¿Dónde se Encuentra la Diversión?..... 50
3.2.6. Desplegar las Herramientas Adecuadas..... 51
3.3. ANALISIS DE CALIDAD, SEGURIDAD Y COSTOS..... 51
3.3.1. Calidad..... 51
3.3.2. Seguridad..... 60
3.3.3. Costos..... 61
3.4. Construcción Casco realidad virtual con Arduino y Godot Engine..... 65
3.4.1. Programa Arduino y el MPU6050..... 66
3.4.2. Instalar librerías..... 67
3.4.3. Conexiones..... 67
3.4.4. Código de Arduino..... 68
3.4.4. Configurando Godot Engine..... 68
3.4.5. Configuración Python en Godot Engine..... 70
CAPITULO IV: CONCLUSIONES Y RECOMENDACIONES..... 72
4.1. Conclusiones..... 72
4.2. Recomendaciones..... 72
BIBLIOGRAFÍAS..... 74
ANEXOS..... 75

INDICE DE FIGURAS

Figura 1 Mecánicas de juegos: 11
Figura 2: Corteza prefrontal.(Psicologia y mente, 2020): 14
Figura 3: Framework D6: 16
Figura 4: Metodología SUM: 18
Figura 5: Calidad: 23
Figura 6. Evaluación Interna, externa y calidad de uso: 24
Figura 7: Usabilidad: .....: 25
Figura 8: Funcionalidad: 26
Figura 9: Confiabilidad 27
Figura 10: Mantenibilidad: 28
Figura 11: Portabilidad: 29
Figura 12: Ventana gráfica: 30
Figura 13: Nodos: 31
Figura 14: Instanciar: 32
Figura 15: Entorno de desarrollo de Arduino: 35
Figura 16: Casco de realidad virtual: 36
Figura 17: MPU-6050: 39
Figura 18: Diagrama de casos de uso: 41
Figura 19: Diagrama navegacional: 44
Figura 20: Nodos e instancias de la escena principal: 45
Figura 21: Menú principal: 45
Figura 22: Casillero Mental: 46
Figura 23: Cálculo mental en ejecución: 47
Figura 24: Mensaje emergente con AdMob: 48
Figura 25: Nodo jugador: 48
Figura 26: Funcionamiento del casco: 65
Figura 27: Dirección de los ejes: 66
Figura 28: Conexión: 67
Figura 29: Formato del código: 68
Figura 30: Ventana del juego: 68

Figura 31: Configuración de nodos: 69
Figura 32: Libreria PythonScript: 70
Figura 33: Casco con pantalla: 71

INDICE DE TABLAS

Tabla: 1 KV-01: Iniciar la Aplicación 42
Tabla: 2 KV-02: Ver Menú principal 42
Tabla: 3 KV-03: Configuración 42
Tabla 4 KV-04: Aplicación 43
Tabla 5 KV-05: Encuesta de usabilidad del sistema 53
Tabla 6 KV-06: Conteo de parámetros PF 55
Tabla 7 KV-07: Ajuste de complejidad 57
Tabla 8 KV-08: Portabilidad 60
Tabla 9 KV-09: Conversión de Puntos de Función 61
Tabla 10 KV-10: Modelo COCOMO 62
Tabla 11 KV-11: Conductores de costo 62
Tabla 12 KV-12: Costo Elaboración del Proyecto 65

1. CAPITULO I: MARCO PRELIMINAR

1.1. Introducción

Hoy en día la tecnología constituye una de las más importantes contribuciones al ser humano que brinda soluciones como por ejemplo a problemas de formación educativa, donde el objetivo es ayudar en el aprendizaje de las personas.

En un mundo en donde la ciencia y la tecnología crece a un ritmo acelerado se crea una demanda educativa cada vez más exigente, sin embargo debemos desarrollar herramientas que nos permitan sobrellevar estas exigencias de la sociedad moderna.

Se ha descubierto que uno de los factores con mayor influencia en las dificultades de los estudiantes para el aprendizaje de la matemática es la motivación. Por su parte, la Gamificación ha surgido en los últimos años como una propuesta para mejorar la interacción, el compromiso de quienes participan en diferentes procesos. En este Proyecto de Grado se pretende aplicar técnicas al Aprendizaje de la Matemática para Sexto de Secundaria en el CEMSE(Centro de Multiservicios Educativos).

Para su desarrollo se utilizara un motor de videojuegos Godot Engine de Software libre, para la Base de Datos se usara la técnica de Singletons (AutoLoad) con Json, los plugins de Adons para publicidad, Arduino y otros similares. Utilizando la metodología Sum que es el consideramos la apropiada.

1.2. Antecedentes

1.2.1. Antecedentes institucionales

CEMSE (Centro de Multiservicios Educativos), es una obra de la Compañía de Jesús en Bolivia, que trabaja desde 1985 al servicio de la educación boliviana, con la finalidad de promover la “igualdad de oportunidades en educación y salud” en especial, de las poblaciones excluidas en desventaja social.

CEMSE fue modelo para la construcción de más de 20 Centros de Multiservicios en todo el país. Actualmente trabajan en La Paz, El Alto, Pucarani, Sucre y Cochabamba, ofreciendo distintos servicios de educación, salud y desarrollo local a través de la dinamización de economías.

Trabajan con padres y madres de familia, maestros y maestras, niños, niñas, adolescentes y jóvenes, responsables populares de salud, brigadas escolares, comunidades productivas y emprendedoras. Cuentan con equipos interdisciplinares de trabajo, compuestos por educadores, médicos, enfermeras, ingenieros, matemáticos, físicos, administradores de empresas, auditores, comunicadores, agrónomos, docentes de aula y facilitadores.

Misión

“Concebir y construir modelos participativos y replicables en educación y salud primaria que promuevan el desarrollo humano de la población más vulnerable de Bolivia, contribuyendo de esta manera al desarrollo local y nacional, apoyando procesos de perfeccionamiento del Estado para superar los índices de pobreza”.

Visión

“El CEMSE (Centro de Multiservicios Educativos) es una institución que promueve la igualdad de oportunidades en educación y salud con buenas prácticas innovadoras y productivas”.

Principios

La educación y la salud con calidad son derechos fundamentales de todos los ciudadanos sin ninguna exclusión.

Las soluciones a los problemas de la población objetivo son integrales, interdisciplinares, significativas y pertinentes, con enfoque de educación y salud para la vida y el desarrollo.

Los derechos humanos, la preservación del medio ambiente y la visión global del mundo, están orientados a la búsqueda de “bien mayor” constituyendo el marco para definir las intervenciones institucionales.

Por el espíritu y naturaleza democrática de la institución, sus intervenciones están enmarcadas en las normas legales vigentes.

El sujeto central de las acciones del CEMSE es la persona con capacidad para construir una sociedad libre, justa e incluyente.

Las intervenciones del CEMSE son "experiencias de vida”, que utilizan como medios las tecnologías, la experimentación, la investigación, la lectura y comunicación, el desarrollo de idiomas universales y la vivencia de valores.

Las intervenciones del CEMSE (Centro de Multiservicios Educativos) integrales, multisectoriales, inter-temporales, inter-generacionales, interculturales, con enfoque de género y derechos.

Valores institucionales

Transparencia en el cumplimiento de acuerdos tanto con la población objetivo, con la población en general y con nuestros financiadores.

Justicia y cultura de paz, siendo facilitadores de una consciencia social para la convivencia inclusiva y el desarrollo armónico de las personas y las comunidades.

Libertad, que desarrolle en las personas y las comunidades, las capacidades de construir sus propios destinos, pensamientos y acciones, en el marco de una convivencia democrática, como base para lograr mejor calidad de vida con dignidad.

Igualdad de oportunidades, que inspira la Vocación de Servicio de la institución bajo el lema: “a mayor pobreza, mayor calidad y calidez en nuestra intervención”.

1.2.2. Antecedente internacional

Titulo: DISEÑO Y DESARROLLO DE UN VIDEOJUEGO EDUCATIVO CON TÉCNICAS DE INTELIGENCIA ARTIFICIAL PARA LA PLATAFORMA ANDROID

Autor: GONZÁLEZ SOTOMAYOR ITAMAR SOLEDAD

Fecha: 2014

Descripción: Este proyecto de tesis, ha tenido como objetivo principal colaborar con los grupos de desarrollo de videojuegos educativos a través del análisis, diseño y desarrollo de un juego didáctico de razonamiento abstracto que consiste en un Laberinto en 3D, para ayudar al desarrollo del pensamiento de niños entre 7 y 11 años.

Metodología: OOHDM

Herramientas: Unity, lenguaje C#

1.2.3. Antecedente nacional

Titulo: TUTOR INTELIGENTE PARA LA OLIMPIADA CIENTÍFICA ESTUDIANTIL PLURINACIONAL BOLIVIANA EN EL ÁREA DE INFORMÁTICA BAJO UN MODELO DE APRENDIZAJE

Autor: YVAN MAYTA QUISPE

Fecha: 2013

Descripción: El objetivo del trabajo es presentar el desarrollo y la aplicación de este tutor inteligente dentro de la enseñanza para la olimpiada científica estudiantil plurinacional en el área de informática bajo un modelo de aprendizaje, el cual es vital importante la interacción con el estudiante de forma dinámica y llamativa, incluimos actividades de acuerdo a las necesidades del estudiante para las olimpiadas.

Metodología: método científico

Herramientas: LENGUAJE PHP

1.2.4. Antecedente local

Título: SOFTWARE EDUCATIVO MULTIMEDIA PARA LA ASIGNATURA DE FISICA – 3ro SECUNDARIA

Autor: HERNAN QUISPE CONURANA

Fecha: 2016

Descripción: Software multimedia educativo para la enseñanza del nivel 3ro de secundaria como herramienta de apoyo de acuerdo a diseño curricular. El cual facilitara el proceso de enseñanza por parte de los docentes a los estudiantes.

Metodología: método propuesto por Galvis

Herramientas: LENGUAJE PHP

1.3. Planteamiento del problema

La Matemática es un tema que todo estudiante de ciencias e ingeniería debería conocer. Pues el conocimiento básico de la matemática nos ayuda a pensar de forma cuantitativa.

Sin embargo, la enseñanza y el aprendizaje de la Matemática en la institución CEMSE(Centro de Multiservicios Educativos) específicamente en Sexto de Secundaria ha presentado ciertas dificultades tanto para los profesores como para los estudiantes y en general, no se ha logrado alcanzar su objetivo de desarrollar este pensamiento de alto nivel que puede ayudarles a resolver problemas de cualquier índole.

Debido a esto, la enseñanza tradicional de la matemática crea algunas veces frustración en el estudiante, la cual hace que la tasa de deserción aumenta en el CEMSE(Centro de Multiservicios Educativos), incumpliendo a su misión y visión.

1.3.1. Problema principal

En la actualidad CEMSE(Centro de Multiservicios Educativos) no cuenta con técnicas didácticos educativos adecuados, para motivar a los estudiantes de Sexto de Secundaria en el Aprendizaje de la Matemática lo que ocasiona que exista deserción de muchos de los estudiantes, por ende perdidas económicas.

1.3.2. Problema secundarios

- √ No existen técnicas adecuadas para lograr la motivación del aprendizaje de la matemática, lo que causa desmotivación a los estudiantes.
- √ Dificultad al desarrollar las capacidades mentales para el aprendizaje de la matemática, por lo que la asimilación de los estudiantes es lenta.
- √ Falta de aplicaciones sencillas y amigables para la motivación del aprendizaje de la matemática, por tanto los estudiantes no aprovechan su tiempo libre de forma productiva.
- √ No se cuenta con tecnología de hardware, que se utilice para la motivación del aprendizaje de la matemática.

1.4. Objetivos

1.4.1. Objetivo general

Desarrollar una Aplicación para la Motivación del Aprendizaje de la Matemática para Sexto de Secundaria, utilizando Técnicas de Gamificación para el Centro de Multiservicios Educativos, como herramienta para motivar al estudiante en el proceso de aprendizaje de conceptos básicos de Matemática de forma eficiente, amigable, agradable y optimizando el tiempo de aprendizaje.

1.4.2. Objetivos específicos

- √ Utilizar las técnicas de Gamificación para lograr la motivación del aprendizaje de la Matemática evitando la desmotivación a los estudiantes.
- √ Establecer las técnicas de Gamificación para desarrollar las capacidades mentales en el aprendizaje de la Matemática para que la asimilación de los

estudiantes sea eficiente.

- √ Diseñar una interfaz de usuario sencilla y amigable para que los estudiantes aprovechen su tiempo.
- √ Utilizar tecnología hardware para la transmisión y recepción de datos procesados por Arduino para el control de la aplicación Gamificada.

1.5. Justificación

1.5.1. Justificación técnica

El uso de la tecnología es una herramienta indispensable para las instituciones educativas tanto publicas y privadas, ello las instituciones educativas deben contar con las herramientas y equipos necesarios.

El CEMSE (Centro de Multiservicios Educativos) cuenta con computadoras, laptops y otro material similar, pese a no contar con muchas computadoras, el proyecto propuesto es de fácil implementación, tanto para usarlo en celulares Android, computadoras con sistema operativo windows o Linux o en web, por eso el proyecto es técnicamente justificable.

1.5.2. Justificación económica

Cuando la aplicación sea utilizada en CEMSE (Centro de Multiservicios Educativos), no será necesario que cada estudiante cuente con un dispositivo de manera individual, pueden utilizarse pocos dispositivos móviles para que los estudiantes realicen las actividades en turnos o grupos. Debemos tomar en cuenta que cada día se incrementa el número de personas que tienen acceso a un dispositivo móvil, a diferencia de hace algunos años, ya no son considerados objetos lujosos y cuentan actualmente con muchas más funcionalidades. Por lo tanto estamos hablando de dispositivos accesibles económicamente. CEMSE (Centro de Multiservicios Educativos) al contar con tales herramientas elevara su calidad de aprendizaje de sus estudiantes, por ende asisten mas estudiantes, aumentado así sus ingresos.

1.5.3. Justificación social

La Gamificación es el uso de técnicas de diseño de juegos en contextos que no

son juegos. Esto significa que se pueden tener algunas bondades de los juegos como la diversión, la tolerancia al fracaso y el compromiso sin necesidad de convertir la actividad completamente en un juego.

En los últimos años, se han realizado estudios, que muestran que aplicar elementos de la Gamificación en el proceso educativo ayuda a aumentar el interés de los estudiantes por los temas estudiados, a disminuir la frustración (Gorris, 1977).

1.6. Metodologías

Para la realización de este trabajo se ha llevado a cabo, en primer lugar, una investigación bibliográfica, en la cual se pretendía encontrar información sobre Gamificación y las posibilidades que ofrece. A través de esta investigación sobre la literatura se ha obtenido la información a partir de la que se ha construido el marco teórico de este trabajo.

Para el desarrollo de la aplicación, se ha empleado la metodología ágil SUM, esta metodología maneja los mismos conceptos de desarrollo de la metodología SCRUM, sin embargo está orientada al desarrollo de videojuegos. Aunque la presente investigación no es juego como tal, se tomara un entorno de videojuegos.

1.7. Herramientas

Las herramientas involucradas en el trabajo son las siguientes:

- √ Lenguaje de programación GdScript.
- √ Software libre Godot Engine con licencia MIT.
- √ Kit de Desarrollo de software de Android(Android SDK).
- √ Programa vectorial de software libre Inkscape.
- √ Programa de manipulación de imágenes de software libre Gimp

1.8. Límites y alcances

1.8.1. Límites

- √ La aplicación estará limitada a funcionar en dispositivos móviles de gama media o superior que necesariamente deberán contar con un sistema operativo Android v.2.3.2 (Gingerbread) o superior.

- √ La aplicación estará limitada a funcionar en sistemas operativos Windows 7 o superior y Linux en distribuciones populares Ubuntu, Debian o similares.
- √ La aplicación cumple el objetivo con la guiá puesta por el profesor, estarán limitados a la calidad de enseñanza que brinde el profesor a cargo.

1.8.2. Alcances

- √ La aplicación que se creará no pretende ser un reemplazo a todo el contenido o actividades que se realizan en aula, sino servir como una herramienta pedagógica para apoyar en el proceso de enseñanza-aprendizaje de la matemática.
- √ La temática de la aplicación a desarrollarse, estará acorde al contenido de desarrollo técnicas de aprendizaje, para la Matemática; dicha temática estará expresada mediante la utilización de elementos de multimedia como: texto, imagen, sonido, y animación, que ayudarán para el mejor aprovechamiento de la asignatura de matemática.

1.9. Aportes

- √ El sistema educativo actual está obsoleta al no ensenar técnicas de aprendizaje. Debido a esto la aplicación permitirá desarrollar técnicas para que los estudiantes del nivel de sexto de secundaria mejorar su rendimiento académico.
- √ Aprender implica estar motivado a hacerlo, al carecer la motivación en la aprendizaje de la matemática en los colegios, la aplicación tiene la finalidad específica de ser utilizado como medio didáctico, para facilitar y mejorar el proceso de aprendizaje.

2. CAPITULO II: MARCO TEÓRICO

2.1. Introducción

En esta sección se va a hacer un estudio del estado actual de la Gamificación y las técnicas de aprendizaje, lo que nos ayudará a entender mejor qué es la Gamificación y cuán extendido está su uso.

2.1.1. Conceptos

Algunos estudios calculan que sólo en Estados Unidos alrededor de treinta y cuatro millones de personas invierten una media de veintidós horas semanales en videojuegos (Siegal, 2014).

Al examinar estos datos es fácil preguntarse qué es lo que hace a las personas dedicar tanto tiempo a algo que realmente no les reporta beneficio alguno e incluso a veces les hace invertir dinero. Y en base a esta pregunta se comenzó a analizar las motivaciones de los jugadores y cómo combinar juegos y aprendizaje en algo que lo llaman Gamificación.

Una de las definiciones de Gamificación más aceptadas es la de Deterding, quien dice que “es el proceso de usar el pensamiento y las mecánicas de juegos en contextos distintos a estos con el fin de resolver problemas y comprometer a los usuarios” (Deterding, Dixon, Khaled y Nacke, 2011).

Con este objetivo en mente, se han tratado de separar los elementos de los juegos que pueden resultar de utilidad para ‘gamificar’ un proceso. Una clasificación de estos elementos propuesta por Werbach y Hunter (Werbach y Hunte, 2014) los divide en tres tipos:

2.1.2. Dinámicas

Estos son los elementos más conceptuales y de alto nivel de un juego. Son los que le dan significado al juego y hacen que toda la experiencia tenga sentido. Entre las dinámicas posibles se pueden encontrar:

- √ Restricciones: Cuáles son las restricciones que impone el juego sobre el mundo real. Algunas de las reglas del juego entran en esta categoría.
- √ Emociones: Cuáles emociones busca inspirar el juego en sus jugadores.
- √ Narrativa: Cuál es la historia que el juego intenta contar, y de qué manera

debe ser entregada al jugador.

- √ Progresión: Cómo el juego hace que el jugador sienta que ha obtenido nuevas habilidades y que el tiempo gastado en el juego ha sido recompensado.
- √ Relaciones: Cómo permite el juego establecer relaciones con otros jugadores o no jugadores.

2.1.3. Mecánicas

Estos elementos son aquellos que permiten realizar las dinámicas y hacen que el juego avance. Entre las mecánicas se pueden encontrar, aunque no están limitadas a:

- √ Retos: El jugador avanza en el juego al superar obstáculos y resolver problemas.
- √ Competencia: La competencia con otros jugadores puede hacer que un jugador reconozca su desempeño en el juego y genere relaciones de rivalidad.
- √ Cooperación: La cooperación entre jugadores permite avances más rápidos y establece relaciones diferentes a la rivalidad.
- √ Retroalimentación: Cuando el jugador recibe respuestas a sus acciones siente que su participación en el juego importa.
- √ Recompensas: Consiste en los premios que obtiene un jugador al cumplir con los objetivos, seguir las reglas, participar en el juego, entre otras. Muchos de los componentes de los juegos están enfocados a recompensar al jugador.
- √ Estados de Victoria: Cuáles son las condiciones que permiten a un jugador progresar de un nivel a otro o avanzar en la narrativa.

2.1.4. Componentes

Estos son los elementos de más bajo nivel que permiten implementar las mecánicas y las dinámicas de una manera específica. Algunos de los componentes utilizados en los juegos son:

- √ Logros: El jugador es recompensado por una acción específica.
- √ Avatares: Es la representación del jugador en el mundo del juego.
- √ Medallas: Son representaciones visuales de los logros obtenidos por el jugador.

- √ Colecciones: Permite al jugador ver el acumulado de logros u otro contenido desbloqueado al que se ha hecho merecedor.
- √ Desbloqueo de Contenido: A medida que el jugador avanza en el desarrollo del juego y adquiere nuevas habilidades, se le permite acceder a contenido que antes era restringido.
- √ Tableros de puntaje: Permite al jugador compararse directamente con otros jugadores.
- √ Niveles: Es una forma de mostrar el avance del jugador. Un jugador de mayor nivel generalmente representa que tiene más habilidad que un jugador de nivel bajo.
- √ Puntos: Es una medida objetiva de qué tanto ha avanzado un jugador, además, funciona como un símbolo de estatus.
- √ Búsquedas: Son retos o problemas puntuales que debe resolver el jugador con el objetivo de recibir alguna recompensa o retroalimentación.

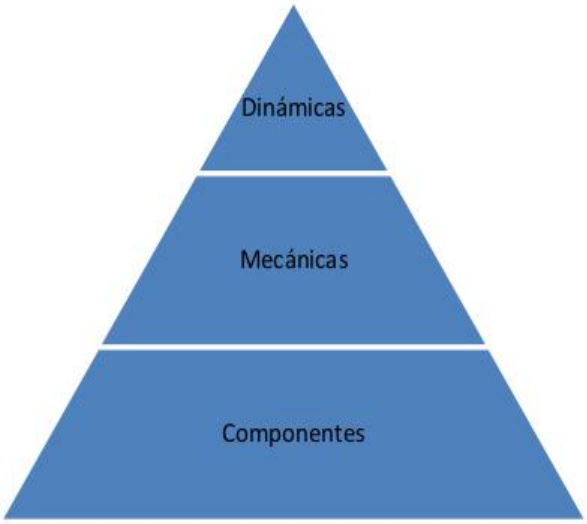


Figura 1: Mecánicas de juegos

2.2. Problemas en el Aprendizaje de la Matemática

Las matemáticas es un tema que todo estudiante de ciencias e ingeniería debería conocer pues son áreas de conocimiento que en esta a la par con la lectura y la

escritura. Sin embargo, los estudiantes no aprecian estos temas debido a distintas razones dependiendo del contexto en el que se les trata de enseñar. Por esta razón se han realizado estudios y diseñado diferentes estrategias para lograr que más estudiantes traten de aprender estos temas y para determinar las causas por las cuales algunos estudiantes presentan dificultades para aprenderlos.

Entre los problemas se pueden encontrar los diferentes estilos de aprendizaje de los estudiantes, los problemas inherentes en los contenidos de los cursos y los problemas motivacionales.

2.2.1. Estilos de Aprendizaje

La clasificación mejor conocida para los estilos de aprendizaje es aquella que los divide en estilo superficial y de profundidad. Los estudiantes adquieren preferencia por alguno de los estilos dependiendo de las capacidades de cada uno, de sus experiencias previas de aprendizaje y de los mecanismos educativos de profesores previos. (Marton y Saljo, 1976).

Aquellos estudiantes que poseen un estilo de aprendizaje superficial se concentran principalmente en la memorización de definiciones y conceptos. Este tipo de aprendizaje funciona bien para ciertas áreas que poseen un cuerpo de conocimiento amplio, por ejemplo la historia o la medicina.

En contraste, los estudiantes que tienen un estilo de profundidad se concentran en conseguir un entendimiento de los temas. (Marton y Saljo, 1976)

Este estilo es especialmente útil cuando es necesario realizar análisis, solución de problemas y generación de nuevo conocimiento. Ambos estilos suelen ser necesarios para el aprendizaje. En la mayoría de las áreas teóricas, primero es necesario memorizar ciertos conocimientos base, para luego utilizar estas bases en conocimientos más profundos.

Esto sin embargo solo aplicar aprendizaje superficial, no logra tener un entendimiento de las matemáticas. No es útil aprender toda la sintaxis y las reglas de un lenguaje de las matemáticas de manera separada y luego estudiar la solución de problemas.

2.2.2. Pensamiento concentrado o difuso

Muchas veces queremos aprender algo pero sin conocer cómo funciona nuestro cerebro. Desde el mismo comienzo del siglo XXI, los neurocientíficos han estado haciendo profundos avances en la comprensión de los dos tipos distintos de redes entre los que va alternando el cerebro: redes para estados de alta atención y redes para estados de reposo, más relajados.

El modo concentrado y el modo difuso: ambos son muy importantes para el aprendizaje. Parece que cambiamos frecuentemente de uno a otro en nuestras actividades diarias. O bien estamos en un modo, o bien en el otro, pero no conscientemente en los dos a la vez. El modo difuso parece capaz de trabajar silenciosamente, en segundo plano, en cosas sobre las que no estamos centrando activamente la atención. A veces también podemos entrar momentáneamente en el modo de pensamiento difuso.

El modo de pensamiento concentrado es esencial para estudiar matemáticas y ciencias. Implica una aproximación directa a la solución de problemas usando enfoques racionales, secuenciales y analíticos. El modo concentrado está asociado a la capacidad para centrar la atención, la cual reside en la corteza prefrontal de tu cerebro, situada justo detrás de tu frente. Centra tu atención en algo y ¡zas!: el modo concentrado se activa, como la luz penetrante del haz de una linterna (Oakley, 2014, p.21).

El modo difuso también es esencial para el aprendizaje matemático. Nos permite adquirir repentinamente una nueva visión de un problema con el que hemos estado peleando y que está asociado a las percepciones a grandes rasgos. El pensamiento de modo difuso tiene lugar cuando relajas tu atención y simplemente dejas vagar la mente. Este desahogo facilita que distintas áreas del cerebro se conecten y den lugar a valiosos hallazgos. A diferencia del modo concentrado, el difuso parece menos asociado a un área cerebral concreta: puedes imaginarlo «difuso» por todo el cerebro. Las inspiraciones en modo difuso a menudo provienen de cavilaciones previas que se han hecho en el modo concentrado. (¡El modo difuso necesita material de construcción!)

El aprendizaje implica una compleja alternancia de procesos neuronales entre

distintas áreas del cerebro, y también entre sus hemisferios. Por lo tanto, ello significa que pensar y aprender es más complicado que simplemente ir alternando entre el modo concentrado y el difuso. (Oakley, 2014, p.22)

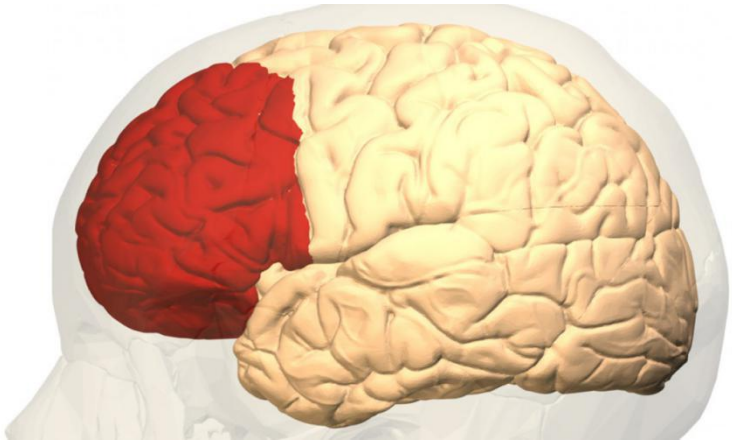


Figura 2: Corteza prefrontal.(Psicología y mente, 2020)

2.2.3. Motivación

La motivación es un concepto abstracto y difícil de medir de manera objetiva, pero que tiene un impacto significativo en el proceso de aprendizaje. Específicamente en el aprendizaje de las matemáticas, tres de los factores principales que afectan la motivación de los estudiantes son: (Feldgen y Glua, 2004)

El estudiante no encuentra una relación entre la profesión elegida y las matemáticas.

El estudiante cree que se necesita cierta habilidad innata para las matemáticas y si no la posee, no podrá aprender sin importar cuánto lo intente.

El estudiante quiera aprender para lograr un objetivo, ya sea lograr el éxito profesional, vencer a sus compañeros académicamente, o impresionar a un profesor o por satisfacción personal.

Para lograr una motivación de forma interna se logra por: (Deci y Ryan, 2004)

√ Competencia: La persona siente que posee habilidad para realizar la tarea requerida.

√ Autonomía: La persona siente que tiene la posibilidad de elegir qué y en qué orden aprende algo.

√ Relación: La persona puede utilizar los conocimientos adquiridos para relacionarse con otras personas

2.3. Framework D6 para el Diseño de Gamificación

El uso metodologías para la Gamificación es nueva, a pesar que no haya, si se desarrollaron frameworks que propone una serie de pasos y herramientas para implementación de la Gamificación.

Este framework, elaborado por Kevin Werbach y Dan Hunter presenta seis pasos a seguir para crear un sistema Gamificado. Los primeros tres pasos están orientados a definir el contexto donde se va a implementar el sistema y a determinar si utilizar Gamificación es una buena solución para el sistema. Los siguientes tres están orientados a describir el sistema Gamificado y a justificar cómo y mediante cuáles mecánicas de juegos el sistema va a mantener motivado al usuario. Los pasos del framework son:

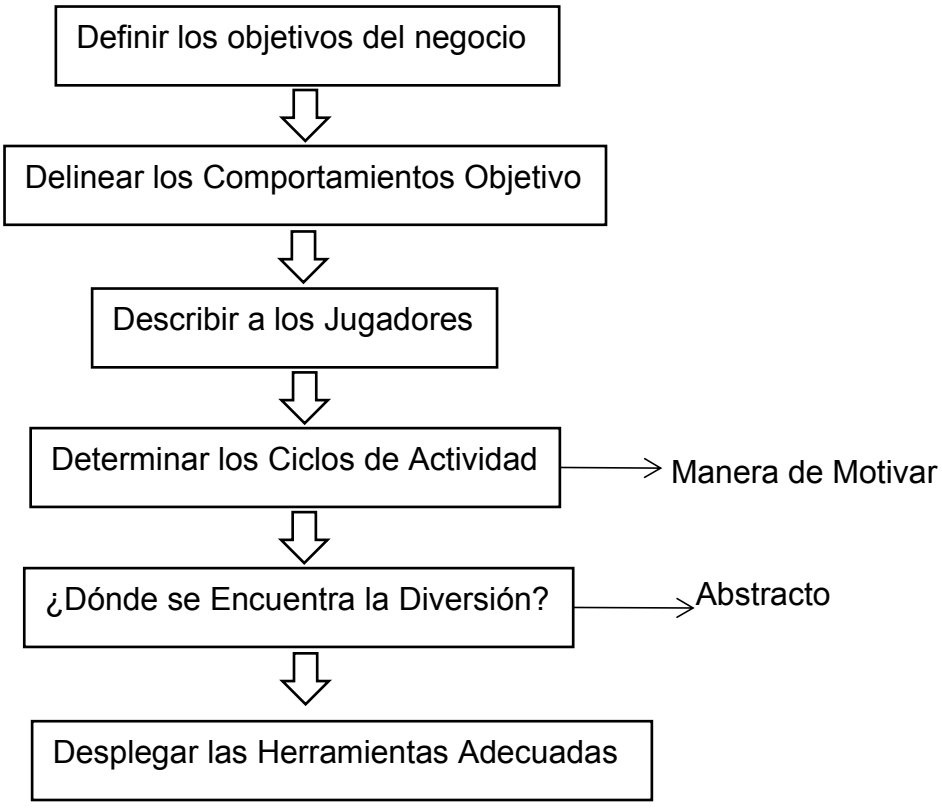


Figura 3: Framework D6

2.3.1. Definir los Objetivos de Negocio:

En este paso se deben definir los objetivos del sistema. Estos objetivos deben resumir por qué se está Gamificando la actividad y qué se espera obtener al hacerlo. En esta sección se hace énfasis en explicar lo que se desea obtener del sistema y no en cómo se va a llevar a cabo.

2.3.2. Delinear los Comportamientos Objetivo

En este paso se deben describir las acciones que se desea los jugadores lleven a cabo al usar el sistema y las métricas que se usarán para determinar si estos comportamientos se están dando. Los comportamientos descritos deben buscar alcanzar los objetivos de negocio y las métricas deben proveer al jugador con retroalimentación de algún tipo para que él sepa si está llevando a cabo las acciones esperadas de su parte.

2.3.3. Describir a los Jugadores

Este paso busca que el diseñador del sistema entienda a sus usuarios (jugadores) potenciales para que en pasos posteriores pueda diseñar la mejor experiencia posible para ellos. En este paso el diseñador ya debe empezar a describir qué tipo de mecánicas de juegos va a utilizar en su sistema, haciendo énfasis en cuáles serían las más efectivas para los jugadores descritos.

2.3.4. Determinar los Ciclos de Actividad

En este paso se explora con más detalle cómo se va a motivar a los jugadores mediante ciclos que comprometan a los usuarios con el sistema y ciclos que permitan al jugador adquirir más experiencia de manera que perciba lo aprendido o logrado desde el momento cuando empezó a utilizar el sistema hasta el momento actual. Es apropiado describir el tipo de retroalimentación que recibirá el jugador, la forma como se hará que nuevos usuarios adopten el sistema y cómo se logrará que usuarios existentes sigan utilizándolo.

2.3.5. ¿Dónde se Encuentra la Diversión?

Este es el paso más abstracto de todos. Asegurarse que el sistema Gamificado sea divertido de utilizar para el usuario es una de las claves de éxito para un proyecto de Gamification. Aquí conviene explorar cómo funcionaría el sistema si no hiciera uso de recompensas extrínsecas (puntos, medallas, logros, entre otras)

con el fin de evitar que el sistema caiga en la trampa de sólo utilizar ese tipo de mecánicas.

2.3.6. Desplegar las Herramientas Adecuadas.

En este paso se explican cuáles son las mecánicas de juego que utilizará el sistema, cómo será la experiencia del jugador al utilizarlo y cómo asegurarse que cumplirá los objetivos.

2.4. Metodología SUM

Las metodologías establecidas para el desarrollo de software no se adaptan a este proceso con garantías de calidad suficientes y no existe en este ámbito un claro planteamiento de cómo afrontar el trabajo. No obstante, se presenta en este proyecto la metodología SUM para videojuegos propuesta en Uruguay, la cual tiene como objetivo desarrollar videojuegos de calidad en tiempo y costo, así como la mejora continua del proceso para incrementar su eficacia y eficiencia. Pretende obtener resultados predecibles, administrar eficientemente los recursos y riesgos del proyecto, y lograr una alta productividad del equipo de desarrollo. SUM fue concebida para que se adapte a equipos multidisciplinarios pequeños, y para proyectos cortos (menores a un año de duración)(Acerenza, 2010).

Lo utilizaremos la metodología SUM simplificada ya que el juego no es extenso, ni tenemos un equipo de desarrollo.

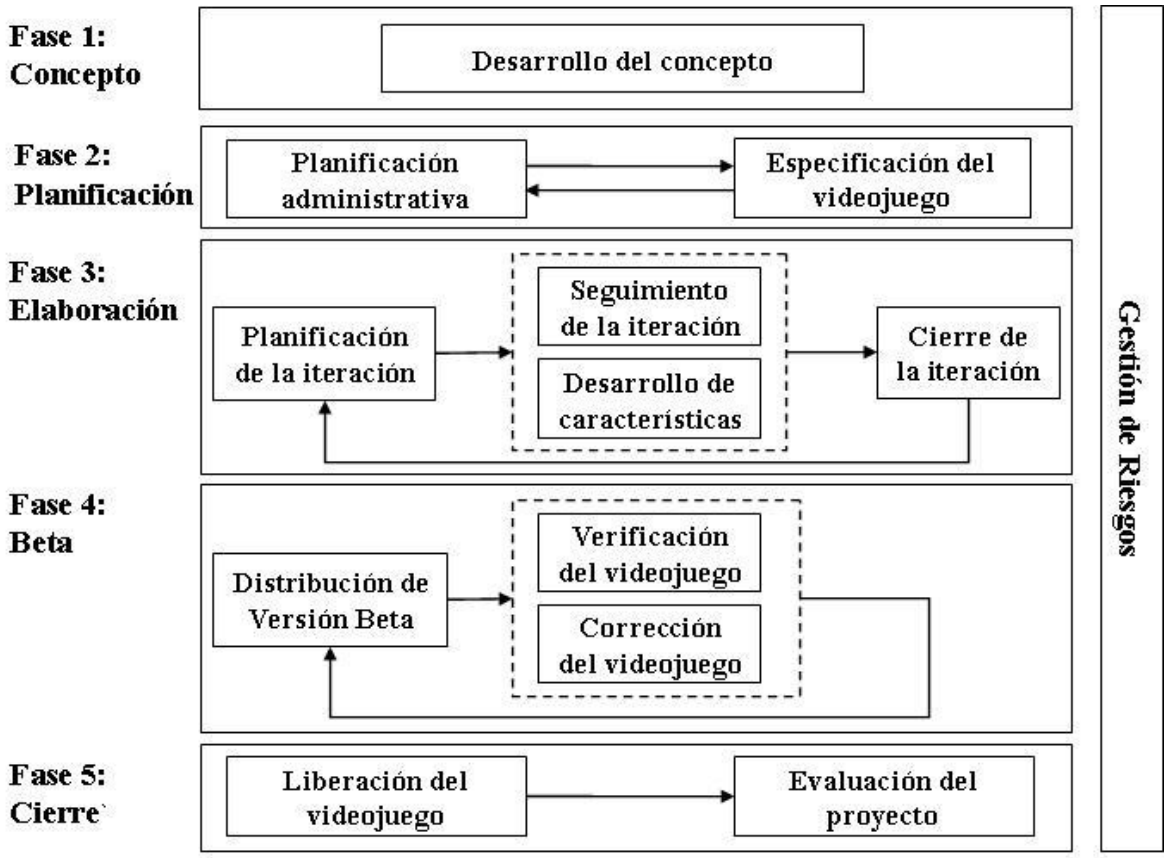


Figura 4: Metodología SUM [Fuente: Acerenza, 2009].

2.4.1. Etapas

2.4.1.1. Fase 1: Concepto

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

2.4.1.2. Fase 2: Planificación

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

La planificación que se obtiene en esta fase es flexible ya que en cada iteración de la fase de elaboración se puede modificar para adaptarse a los cambios y reflejar la situación actual del proyecto.

2.4.1.3. Fase 3: Elaboración

El objetivo de esta fase es implementar el videojuego, para ello se trabaja en forma iterativa e incremental para lograr una versión ejecutable del videojuego al finalizar cada iteración. Éstas se dividen en tres etapas, en la primera se planifican los objetivos a cumplir, las métricas a utilizar en el seguimiento, las características a implementar y las tareas necesarias para ello. En la segunda se desarrollan las características planificadas a través de la ejecución de las tareas que la componen. Al mismo tiempo se realiza el seguimiento para mantener la visión y el control de la iteración en base a los objetivos planeados. La tercera y última implica la evaluación del estado del videojuego y de lo ocurrido en el transcurso de la iteración para actualizar el plan de proyecto respecto a la situación actual. Con esta forma de trabajo se puede evaluar el avance del proyecto, lo cual permite realizar cambios a tiempo y tomar decisiones para cumplir con las tareas. Además, la experiencia adquirida permite mejorar la forma de trabajo en cada iteración y aumentar la productividad.

2.4.1.4. Fase 4: Beta

La fase tiene como objetivo evaluar y ajustar distintos aspectos del videojuego como por ejemplo gameplay, diversión, curva de aprendizaje y curva de dificultad,

además de eliminar la mayor cantidad de errores detectados. Se trabaja en forma iterativa liberando distintas versiones del videojuego a verificar. Para ello primero se distribuye la versión beta del videojuego a verificar y se determinan los aspectos a evaluar. Mientras esta se verifica, se envían reportes con los errores o evaluaciones realizadas. Estos reportes son analizados para ver la necesidad de realizar ajustes al videojuego. Se puede optar por liberar una nueva versión del videojuego para verificar una vez que se realizan los ajustes. Y el ciclo termina cuando se alcanza el criterio de finalización establecido en el plan del proyecto.

2.4.1.5. Fase 5: Cierre

Esta fase tiene como objetivos entregar la versión final del videojuego al cliente según las formas establecidas y evaluar el desarrollo del proyecto. Para la evaluación se estudian los problemas ocurridos, los éxitos conseguidos, las soluciones halladas, el cumplimiento de objetivos y la certeza de las estimaciones. Con las conclusiones extraídas se registran las lecciones aprendidas y se plantean mejoras a la metodología.

2.4.2. Ventajas y Desventajas

2.4.2.1. Ventajas

- √ Facilidad de adopción por el equipo de desarrollo.
- √ Facilita la administración del proyecto.
- √ Aplicable en equipos de desarrollo grandes.
- √ Fácil aplicabilidad en proyectos que se basan en modificaciones pequeñas de productos ya realizados (secuelas de juegos).

2.4.2.2. Desventajas

- √ No responde bien ante cambios.
- √ Existen problemas de comunicación e interpretación.
- √ Ningún objeto puede ser considerado completo hasta que todas las dependencias de ese objeto hayan sido construidas.
- √ Cuando hay departamentos de desarrollo que dependen uno de otro se

generan situaciones de cuello de botella.

2.5. Costo y/o beneficio

2.5.1. Método de COCOMO

Entre los distintos métodos de estimación de costes de desarrollo de software, el modelo COCOMO (Constructive Cost Model) desarrollado por Barry M. Boehm, se engloba en el grupo de los modelos algorítmicos que tratan de establecer una relación matemática la cual permite estimar el esfuerzo y tiempo requerido para desarrollar un producto.

Por un lado COCOMO define tres modos de desarrollo:

Orgánico: proyectos relativamente sencillos, menores de 5000 LDC líneas de código, en los cuales se tiene experiencia de proyectos similares y se encuentran en entornos estables.

Semi-acoplado: proyectos intermedios en complejidad y tamaño, donde la experiencia en este tipo de proyectos es variable, y las restricciones intermedias.

Empotrado: proyectos bastantes complejos, en los que apenas se tienen experiencia y se engloban en un entorno de gran innovación técnica. Además se trabaja con unos requisitos muy restrictivos y de gran volatilidad.

Y por otro lado existen diferentes modelos que define COCOMO:

Modelo básico: Se basa exclusivamente en el tamaño expresado en LDC.

Modelo intermedio: Además del tamaño del programa incluye un conjunto de medidas subjetivas llamadas conductores de costes.

Modelo avanzado: Incluye todo lo del modelo intermedio además del impacto de cada conductor de coste en las distintas fases de desarrollo.

Para nuestro caso el modelo intermedio será el que usaremos, dado que realiza las estimaciones con bastante precisión.

Así pues las fórmulas serán las siguientes:

E = Esfuerzo = a KLDC e * FAE (persona x mes)

T = Tiempo de duración del desarrollo = c Esfuerzo d (meses)

P= Personal = E/T (personas)

2.6. METRICAS DE CALIDAD

2.6.1. ISO/IEC 9126

Esta norma Internacional fue publicada en 1992, la cual es usada para la evaluación de la calidad de software, llamado “Information technology-Software product evaluation-Quality characteristics and guidelines for their use”; o también conocido como ISO 9126 (o ISO/IEC 9126). Este estándar describe 6 características generales: Funcionalidad, Confiabilidad, Usabilidad, Eficiencia, Mantenibilidad, y Portabilidad.

La norma ISO/IEC 9126 permite especificar y evaluar la calidad del software desde diferentes criterios asociados con adquisición, requerimientos, desarrollo, uso, evaluación, soporte, mantenimiento, aseguramiento de la calidad y auditoria de software. Los modelos de calidad para el software se describen así:

Calidad interna y externa: Especifica 6 características para calidad interna y externa, las cuales, están subdivididas. Estas divisiones se manifiestan externamente cuando el software es usado como parte de un sistema Informático, y son el resultado de atributos internos de software.

Calidad en uso: Calidad en uso es el efecto combinado para el usuario final de las 6 características de la calidad interna y externa del software. Especifica 4 características para la calidad en uso.

Al unir la calidad interna y externa con la calidad en uso se define un modelo de evaluación más completo, se puede pensar que la usabilidad del modelo de calidad externa e interna pueda ser igual al modelo de calidad en uso, pero no, la usabilidad es la forma como los profesionales interpretan o asimilan la funcionabilidad del software y la calidad en uso se puede asumir como la forma que lo asimila o maneja el usuario final. Si se unen los dos modelos, se puede definir que los seis indicadores del primer modelo tienen sus atributos y el modelo de calidad en uso sus 4 indicadores pasarían hacer sus atributos, mirándolo gráficamente quedaría así:

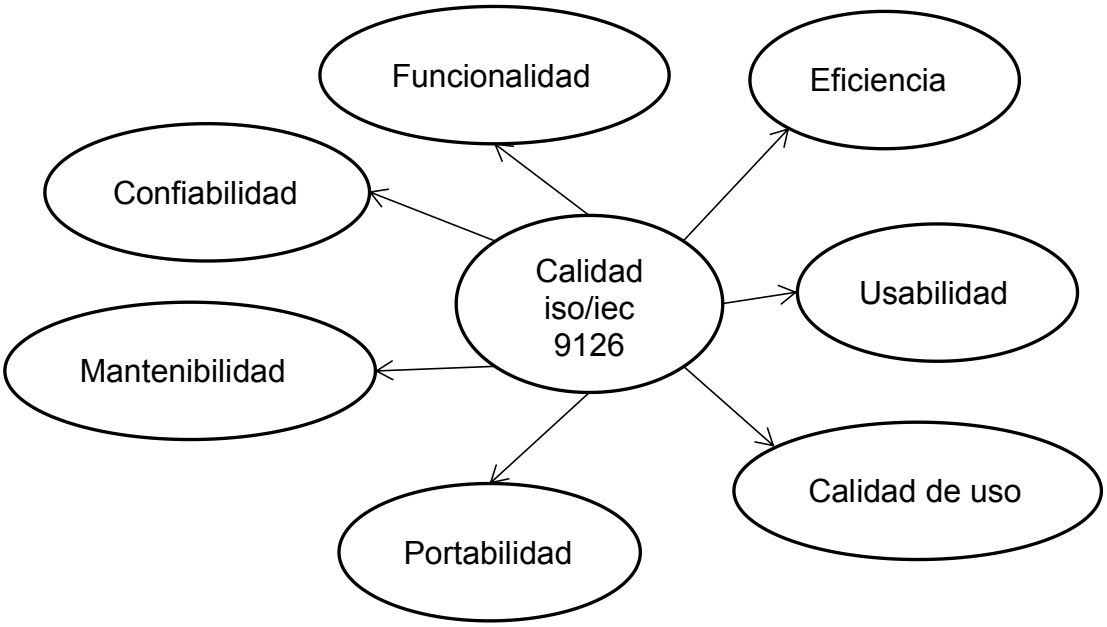


Figura 5: Calidad [Fuente: ISO/IEC, 2007]

Se establecen categorías para las cualidades de la calidad externa e interna y calidad en uso del software, teniendo en cuenta estos 7 indicadores (funcionalidad, confiabilidad, utilidad, eficiencia, capacidad de mantenimiento, portabilidad y calidad en uso), que se subdividen a su vez en varios indicadores; estas se pueden medir por métrica interna o externa.

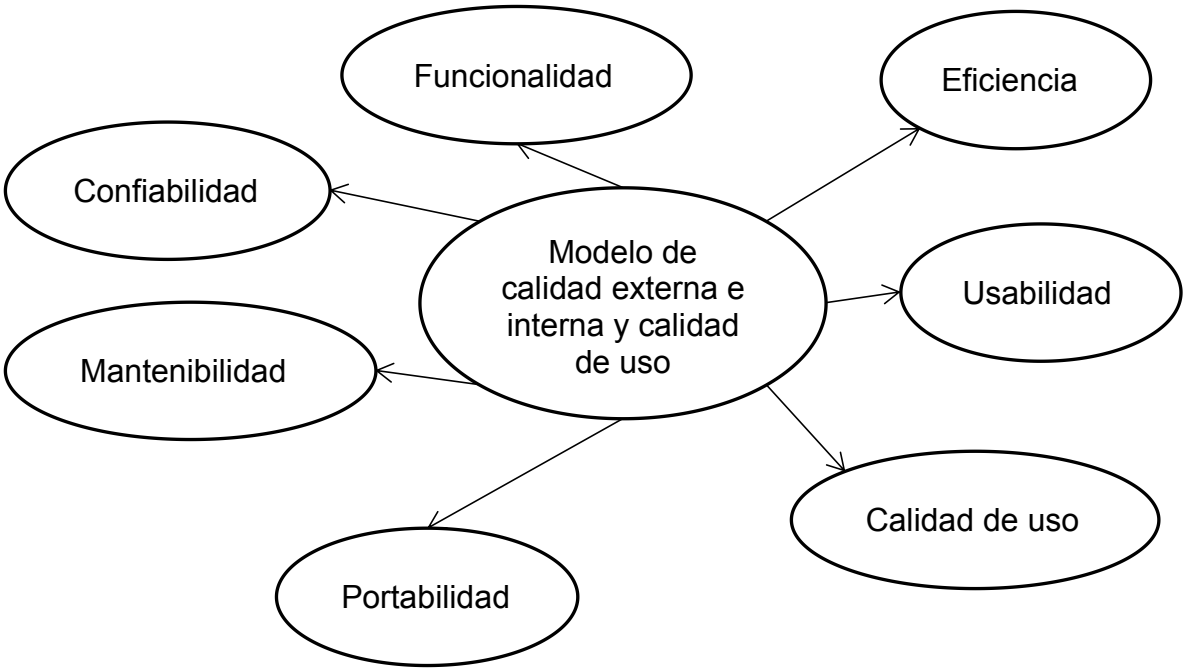


Figura 6: Evaluación Interna, externa y calidad de uso ISO/IEC 9126 [Fuente: ISO/IEC, 2007]

Las definiciones se dan para cada característica y sub-característica de calidad del software que influye en la calidad. Para cada característica y sub-característica, la capacidad del software es determinada por un conjunto de atributos internos que pueden ser medidos. Las características y sub-características se pueden medir externamente por la capacidad del sistema que contiene el software.

2.6.1.1. Usabilidad

La usabilidad es la capacidad del software de ser entendido, aprendido, y usado en forma fácil y atractiva. Algunos criterios de funcionalidad, fiabilidad y eficiencia afectan la usabilidad, pero para los propósitos de la ISO/IEC 9126 ellos no clasifican como usabilidad. La usabilidad está determinada por los usuarios finales y los usuarios indirectos del software, dirigidos a todos los ambientes, a la preparación del uso y el resultado obtenido.

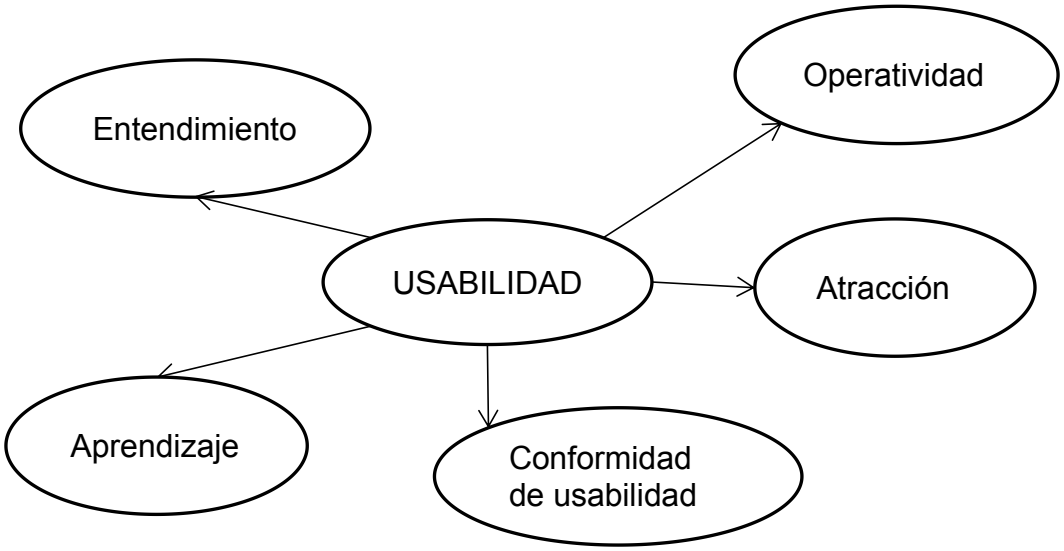


Figura 7: Usabilidad [Fuente: ISO/IEC, 2007]

La usabilidad se divide en 5 criterios:

Entendimiento: La capacidad que tiene el software para permitir al usuario entender si es adecuado, y de una manera fácil como ser utilizado para las tareas y las condiciones particulares de la aplicación. En este criterio se debe tener en cuenta la documentación y de las ayudas que el software entrega.

Aprendizaje: La forma como el software permite al usuario aprender su uso. También es importante considerar la documentación.

Operatividad: La manera como el software permite al usuario operarlo y controlarlo.

Atracción: La presentación del software debe ser atractiva al usuario. Esto se refiere a las cualidades del software para hacer más agradable al usuario, ejemplo,

el diseño gráfico.

Conformidad de uso: La capacidad del software de cumplir los estándares o normas relacionadas a su usabilidad.

2.6.1.2. Funcionalidad

Funcionalidad es la capacidad del software de cumplir y proveer las funciones para satisfacer las necesidades explícitas e implícitas cuando es utilizado en condiciones específicas. A continuación se muestra la característica de Funcionalidad y las subcaracterísticas que cubre:

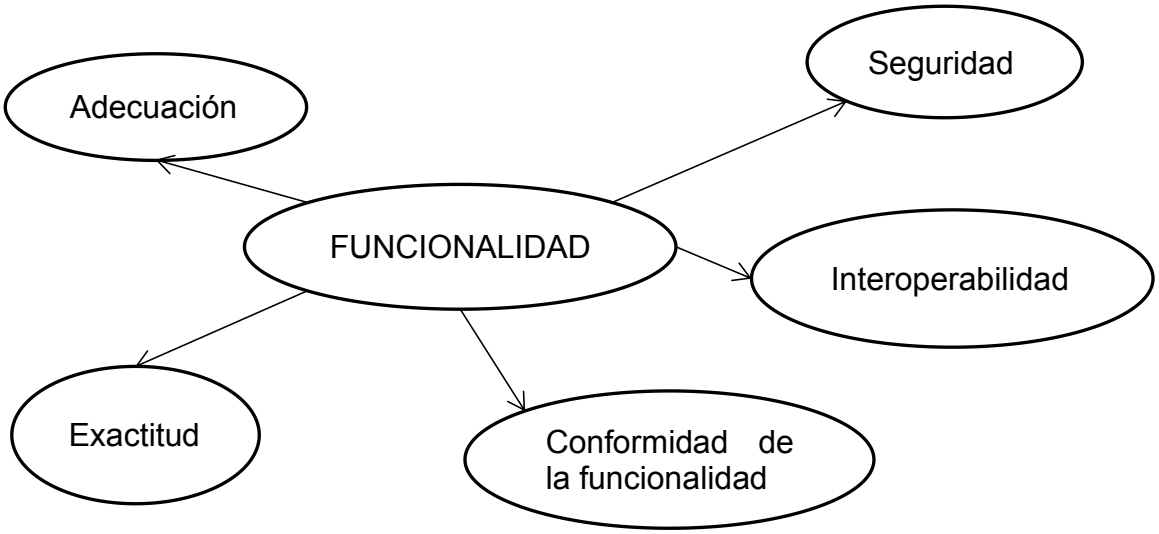


Figura 8: Funcionalidad [Fuente: ISO/IEC, 2007]

La funcionalidad se divide en 5 criterios:

Adecuación: La capacidad del software para proveer un adecuado conjunto de funciones que cumplan las tareas y objetivos especificados por el usuario.

Exactitud: La capacidad del software para hacer procesos y entregar los resultados solicitados con precisión o de forma esperada.

Interoperabilidad: La capacidad del software de interactuar con uno o más sistemas específicos.

Seguridad: La capacidad del software para proteger la información y los datos de manera que los usuarios o los sistemas no autorizados no puedan acceder a ellos

para realizar operaciones, y la capacidad de aceptar el acceso a los datos de los usuarios o sistemas autorizados.

Conformidad de la funcionalidad: La capacidad del software de cumplir los estándares referentes a la funcionalidad.

2.6.1.3. Confiabilidad

La confiabilidad es la capacidad del software para asegurar un nivel de funcionamiento adecuado cuando es utilizando en condiciones específicas. En este caso la confiabilidad se amplía en sostener un nivel especificado de funcionamiento y no una función requerida.

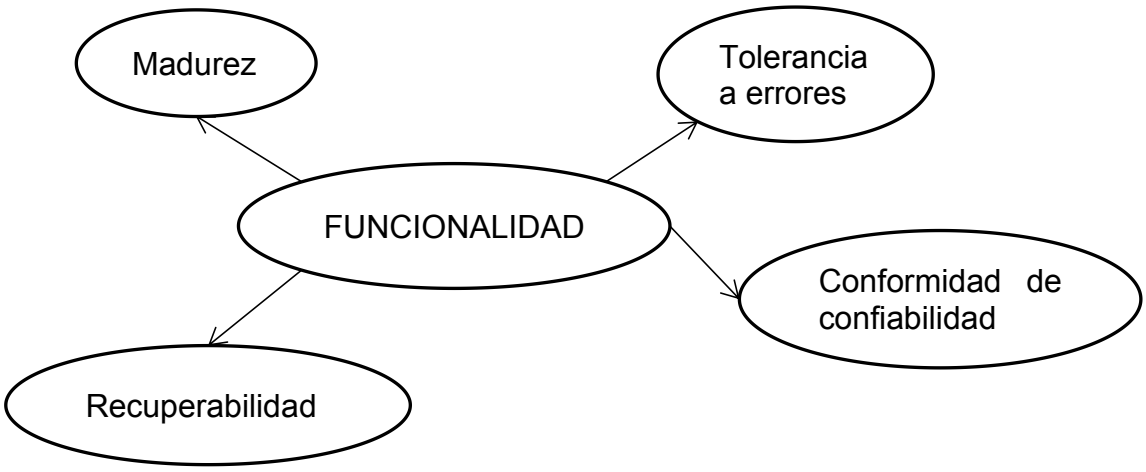


Figura 9: Confiabilidad [Fuente: ISO/IEC, 2007]

La confiabilidad se divide en 4 criterios:

Madurez: La capacidad que tiene el software para evitar fallas cuando encuentra errores. Ejemplo, la forma como el software advierte al usuario cuando realiza operaciones en la unidad de diskett vacía, o cuando no encuentra espacio suficiente el disco duro donde esta almacenando los datos.

Tolerancia a errores: La capacidad que tiene el software para mantener un nivel de funcionamiento en caso de errores.

Recuperabilidad: La capacidad que tiene el software para restablecer su funcionamiento adecuado y recuperar los datos afectados en el caso de una falla.

Conformidad de la fiabilidad: La capacidad del software de cumplir a los estándares o normas relacionadas a la fiabilidad.

2.6.1.4. Mantenibilidad

La mantenibilidad es la cualidad que tiene el software para ser modificado. Incluyendo correcciones o mejoras del software, a cambios en el entorno, y especificaciones de requerimientos funcionales.

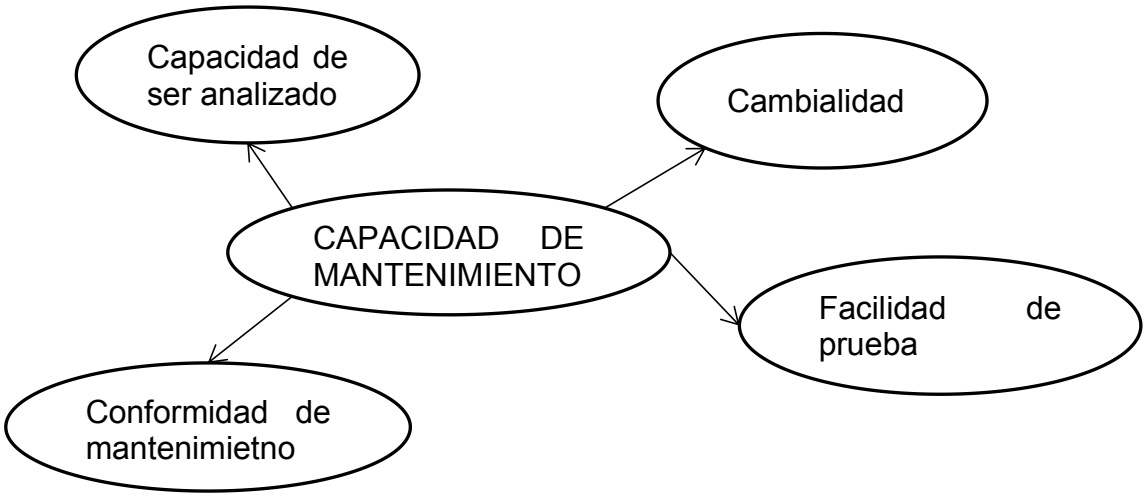


Figura 10: Mantenibilidad [Fuente: ISO/IEC, 2007]

El mantenimiento se divide en 5 criterios:

Capacidad de ser analizado: La forma como el software permite diagnósticos de deficiencias o causas de fallas, o la identificación de partes modificadas.

Cambiabilidad: La capacidad del software para que la implementación de una modificación se pueda realizar, incluye también codificación, diseño y documentación de cambios.

Estabilidad: La forma como el software evita efectos inesperados para modificaciones del mismo.

Facilidad de prueba: La forma como el software permite realizar pruebas a las modificaciones sin poner el riesgo los datos.

Conformidad de facilidad de mantenimiento: La capacidad que tiene el software para cumplir con los estándares de facilidad de mantenimiento.

2.6.1.5. Portabilidad

La capacidad que tiene el software para ser trasladado de un entorno a otro.

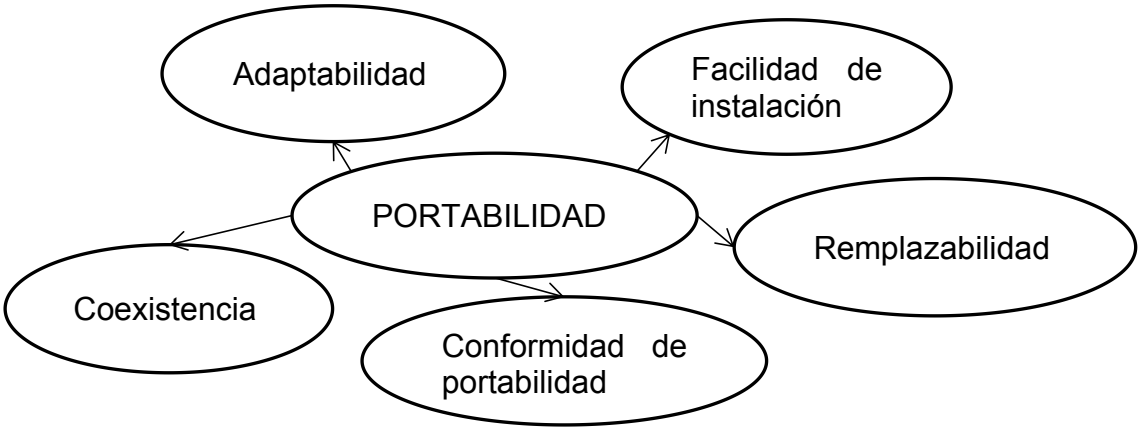


Figura 11: Portabilidad [Fuente: ISO/IEC, 2007]

La usabilidad se divide en 5 criterios:

Adaptabilidad: Es como el software se adapta a diferentes entornos especificados (hardware o sistemas operativos) sin que implique reacciones negativas ante el cambio. Incluye la escalabilidad de capacidad interna (Ejemplo: Campos en pantalla, tablas, volúmenes de transacciones, formatos de reporte, etc.).

Facilidad de instalación: La facilidad del software para ser instalado en un entorno específico o por el usuario final.

Coexistencia: La capacidad que tiene el software para coexistir con otro o varios software, la forma de compartir recursos comunes con otro software o dispositivo.

Reemplazabilidad: La capacidad que tiene el software para ser remplazado por otro software del mismo tipo, y para el mismo objetivo. Ejemplo, la reemplazabilidad de una nueva versión es importante para el usuario, la propiedad de poder migrar los datos a otro software de diferente proveedor.

Conformidad de portabilidad: La capacidad que tiene el software para cumplir con los estándares relacionados a la portabilidad.

2.7. Godot Engine

Godot Engine es un motor de videojuegos multiplataforma con múltiples características para crear juegos 2D y 3D desde una interfaz unificada. Él provee un conjunto exhaustivo de herramientas comunes para que los usuarios puedan enfocarse en crear juegos sin tener que reinventar la rueda. Juegos que pueden exportarse en un sólo clic a numerosas plataformas, incluyendo las principales plataformas de escritorio (Linux, macOS, Windows), móviles (Android, iOS) y basadas en la web (HTML5).

Godot es completamente gratuito y de código abierto bajo la licencia permisiva del MIT. Sin condiciones, sin regalías, nada. Los juegos de los usuarios son suyos, hasta la última línea del código del motor. El desarrollo de Godot es totalmente independiente y dirigido por la comunidad, lo que permite a los usuarios ayudar a dar forma a su motor para que coincida con sus expectativas. Está respaldado por Software Freedom Conservancy sin fines de lucro.

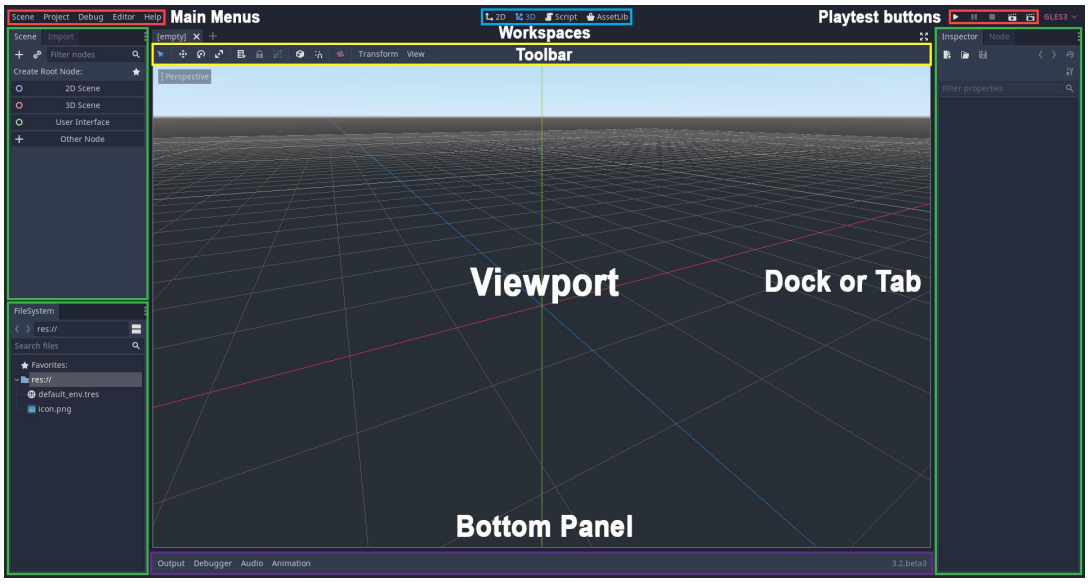


Figura 12: Ventana gráfica
Fuente: godotengine,2020

2.7.1. Escenas y nodos

Los nodos son elementos fundamentales para crear un juego. Un nodo puede realizar una variedad de funciones especializadas. Sin embargo, cualquier nodo utilizado siempre tiene los siguientes atributos:

√ Tiene un nombre.

√ Tiene propiedades editables.

√ Puede recibir una llamada a una función (callback) para procesar cada fotograma.

√ Se puede extender (para tener más funciones).

√ Se puede añadir a otro nodo como un hijo.

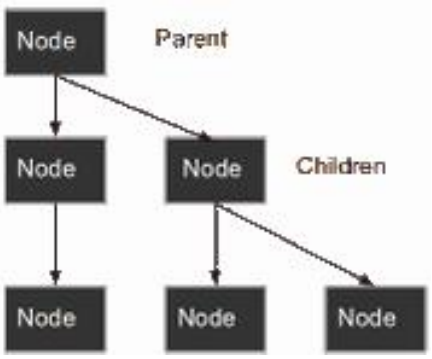


Figura 13: Nodos
Fuente: godotengine, 2020

Una escena se compone de un grupo de nodos organizados jerárquicamente (en forma de árbol). Además, una escena:

√ siempre tiene un solo nodo raíz.

√ Se puede guardar en disco y cargarse de nuevo.

√ Se puede Instanciar (explicado más adelante).

Ejecutar un juego significa ejecutar una escena. Un proyecto puede contener varias escenas, pero para que comience el juego, se debe seleccionar una de ellas como la escena principal.

Básicamente, el editor de Godot es un editor de escena. Tiene muchas herramientas para editar escenas 2D y 3D, así como también interfaces de usuario, pero el editor se basa en el concepto de edición de una escena y los

nodos que la componen.

2.7.2. Instanciar

Uno de los puntos fuertes de la instanciación de escenas, es que funciona como un excelente lenguaje de diseño. Es lo que distingue a Godot de otros motores. Godot se diseñó desde su base en torno a este concepto.

Cuando haces juegos con Godot, el enfoque recomendado es descartar los patrones de diseño comunes, como MVC o diagramas de Entidad-Relación, en su lugar piensa en tus escenas en un modo más natural. Comienza imaginando los elementos visibles de tu juego, los que pueden ser nombrados no sólo por un programador, sino por cualquiera.

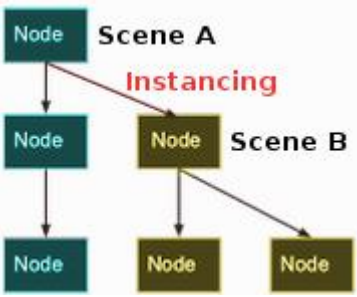


Figura 14: Instanciar
Fuente: godotengine,2020

Instanciar tiene mucha utilidad. A primera vista, con instanciar tienes:

√ La habilidad de dividir escenas y hacerlas más fácil de manejar.

√ Una herramienta para gestionar y editar múltiples instancias de nodo a la vez.

√ Una manera de organizar flujos de juego complejos, incluso para UI (en godot, los elementos UI son nodos también).

2.7.3. Scripting

Los lenguajes "principales" en Godot son GDScript y VisualScript. La razón principal para elegirlos es su nivel de integración con Godot, ya que esto hace que la experiencia sea más agradable; ambos tienen una integración con el editor muy fluida, mientras que con C# y C++ se necesita editarlos en un IDE separado. Si eres un gran fan de los lenguajes de tipado estático, utiliza C# y C++ en su lugar.

2.7.3.1. GDScript

Es el lenguaje principal utilizado en Godot. Su uso tiene algunos puntos positivos en comparación con otros idiomas debido a su alta integración con Godot:

- √ Es simple, elegante, y diseñado para que resulte muy familiar a los usuarios de otros idiomas como Lua, Python, Squirrel, etc.
- √ Carga y compila a toda velocidad.
- √ Es un placer trabajar con la integración del editor, con completado de código para nodos, señales y muchos otros elementos relacionados con la escena que se está editando.
- √ Tiene tipos de vectores propios (tales como Vector, transforms, etc.), haciéndolo eficiente para el uso intensivo del álgebra lineal.
- √ Soporta múltiples hilos de forma tan eficiente como los lenguajes de tipado estático - una de las limitaciones que nos hizo evitar VMs como Lua, Squirrel, etc.
- √ No utiliza ningún recolector de basura, por lo que intercambia un poco de automatización (de todos modos, la mayoría de objetos son pasados por referencia), por determinismo.
- √ Su naturaleza dinámica hace que sea fácil optimizar secciones de código en C++ (vía GDNative) si se requiere más rendimiento, todo ello sin recompilar el motor.

2.8. Arduino

Arduino es una plataforma electrónica de hardware libre basada en una placa con un micro controlador. Con software y hardware flexibles y fáciles de utilizar, Arduino ha sido diseñado para adaptarse a las necesidades de todo tipo de público, desde aficionados, hasta expertos en robótica o equipos electrónicos (Arduino, 2015).

Ante todo y sobre todo es un micro controlador, es decir un ordenador completo integrado en un chip, con su CPU, memoria de programa, memoria de datos y circuitos para el control de periféricos.

El micro controlador necesita para su correcto funcionamiento, de algunos

circuitos auxiliares y complementos tales como:

• La entrada de alimentación

• El oscilador de trabajo

• Circuito de RESET

• La conexión USB

• Los accesos a las líneas de entrada y salida, etc

2.8.1. Características

La Arduino Uno es una board basada en un micro controlador Atmega328. Tiene 14 pines de entrada/salida digital (de los cuales 4 pueden ser utilizados para salidas PWM), 6 entradas análogas, un resonador cerámico de 16 MHz, un conector para USB tipo hembra, un Jack para fuente de Poder, un conector ICSP y un botón reset.

Tiene todo lo necesario para manejar el controlador, simplemente conectamos al computador por medio del cable USB o una fuente de poder externa, que puede ser un adaptador AC-DC o una batería, cabe aclarar que si se alimenta a través del cable USB en el ordenador no es necesario una fuente externa.(Arduino, 2016).

√ Micro controlador: ATmega328

√ Voltaje Operativo: 5v

√ Voltaje de Entrada (Recomendado): 7 – 12 v

√ Pines de Entradas/Salidas Digital: 14 (De las cuales 6 son salidas PWM)

√ Pines de Entradas Análogas: 6

√ Memoria Flash: 32 KB (ATmega328) de los cuales 0,5 KB es usado por Bootloader.

√ SRAM: 2 KB (ATmega328)

√ EEPROM: 1 KB (ATmega328)

√ Velocidad del Reloj: 16 MHZ.

2.8.2. Programación

El ATmega328 en el Arduino Uno viene precargado con un gestor de arranque

(bootloader) que permite cargar nuevo código sin necesidad de un programador por hardware externo. Se comunica utilizando el protocolo STK500 original (referencia, archivo de cabecera C).

También puede evitarse el gestor de arranque y programar directamente el micro controlador a través del puerto ICSP (In Circuit Serial Programming).

Figura 15: Entorno de desarrollo de Arduino

Fuente: Elaboración Propia

2.9. Realidad virtual

Un casco de realidad virtual, también llamado gafas de realidad virtual, visor o HMD (del inglés head-mounted display), es un dispositivo de visualización similar a un casco, que permite reproducir imágenes creadas por ordenador sobre una pantalla muy cercana a los ojos o proyectando la imagen directamente sobre la retina de los ojos. En este segundo caso el casco de realidad virtual recibe el nombre de monitor virtual de retina.

Debido a su proximidad con los ojos, el casco de realidad virtual consigue que las imágenes visualizadas resulten mucho mayores que las percibidas por pantallas normales, y permiten incluso englobar todo el campo de visión del usuario. Gracias a que el casco se encuentra sujeto a la cabeza, éste puede seguir los movimientos del usuario, consiguiendo así que éste se sienta integrado en los ambientes creados por ordenador.

Figura 16: Casco de realidad virtual

Fuente: Virtual Reality headset by Karajko CAD

2.9.1. Tipos

Según reproduzcan las imágenes sobre un ojo o sobre los dos, los HMD pueden ser:

√ Monocular: las imágenes sólo se reproducen sobre un ojo. Técnicamente es un HMD pero no es para realidad virtual. Es el caso de las Google Glass.

√ Binocular: las imágenes se reproducen sobre los dos ojos, obteniendo así una imagen estereoscópica.

Por otro lado, cabe distinguir también:

√ Cascos o gafas de realidad virtual: ocupan el campo de visión del usuario de modo que no tiene percepción del entorno que lo rodea, permitiendo así la completa inmersión de éste en una realidad virtual, ya que sólo percibirá las imágenes creadas por ordenador y reproducidas sobre la pantalla.

√ Cascos o gafas de realidad aumentada o realidad mixta: conocidos también como HMD óptico (u OHMD) permiten al usuario ver todo el entorno que lo

rodea e introducen en éste objetos virtuales o información, produciendo así lo que se conoce como realidad aumentada o realidad mixta. Dentro de esta categoría se incluyen las gafas inteligentes, cuyo principal uso es mostrar información disponible para los usuarios de teléfonos inteligentes sin utilizar las manos.

Por último, según su operatividad, se pueden distinguir:

- √ Gafas de realidad virtual móvil: realmente son carcasas, que no tienen pantalla propia ni procesador sino que están preparadas para albergar un teléfono móvil, en el cual se reproducirán las imágenes. Ejemplos: Gear VR de Samsung, Cardboard de Google, y muchas otras de distintos fabricantes.
- √ Gafas de realidad virtual sin procesador: incluyen pantalla propia y sensores pero se conectan a un aparato externo (típicamente un ordenador personal) para recibir las imágenes. Ejemplos: Oculus Rift, PlayStation VR, HTC Vive...
- √ Gafas de realidad virtual autónomas: son las que incluyen todos los componentes necesarios, como la carcasa, pantalla, sensores y procesador. Ejemplo: Microsoft Hololens y otras en desarrollo como las Project Alloy de Intel, las Daydream Standalone de Qualcomm y Google, o las Exynos VR de Samsung.

2.9.2. Características

Existen varios conceptos clave en la tecnología que emplean los cascos de realidad virtual. Entre ellos podemos destacar:

Resolución de pantalla: es un parámetro muy importante pues de ella depende mayormente la definición de la imagen percibida por el usuario del HMD. Una resolución típica a día de hoy (principios de 2016) son los 1080x1200 píxeles para cada ojo del Oculus Rift y del HTC Vive.

Campo de visión (en inglés field of view, FoV): es la amplitud del campo visual del usuario que es ocupada por la imagen virtual. Cuanto mayor sea, mejor será la sensación de inmersión. El Oculus Rift DK2 por ejemplo ofrece un campo de visión de 100º.

Latencia de seguimiento (head tracking latency): es el tiempo que transcurre

entre momento en que el usuario mueve su cabeza y aquel en el que la imagen mostrada se reajusta a ese movimiento. Los fabricantes intentan reducirla al mínimo pues una excesiva latencia puede producir mareos en los usuarios, además de un menor realismo. PlaySation VR registra a una latencia de 18 ms.

Refresco de pantalla (refresh rate): el número de imágenes mostradas por segundo. A partir de 60 Hz se considera un buen ratio. Así por ejemplo, el visor HTC Vive Pre y el Oculus Rift CV1 funcionan a 90 Hz, mientras el PlayStation VR alcanza los 120 Hz.

Seguimiento de orientación (head tracking o rotational tracking): mediante sensores internos (giroscopio, acelerómetro, magnetómetro) el HMD detecta hacia dónde está orientada la cabeza del usuario.

Seguimiento de posición (positional tracking): también conocido como posicionamiento absoluto, se logra mediante un sensor, normalmente externo a las propias gafas, que detecta dónde está situada exactamente la cabeza del usuario y cualquier cambio que se produzca en esa posición. Es una característica que solo incorporan los HMD más avanzados.

Seguimiento ocular (eye tracking): mediante unos sensores infrarrojos dentro del casco se captan los movimientos del ojo. Esto permite cosas como replicar los movimientos de tus ojos en tu avatar virtual, o provocar reacciones de otros personajes según la manera en la que los miras. Pionero de esta funcionalidad es el modelo FOVE VR.

Visión estereoscópica: característica presente en casi todos los aparatos de realidad virtual, que mostrando una imagen ligeramente diferente a cada ojo permite visualizar el entorno en tres dimensiones.

Efecto rejilla (screen-door effect): es un efecto visual que sucede en pantallas cuando las líneas que separan los píxeles de la misma se vuelven visibles en la imagen proyectada. El resultado es similar al de mirar a través de una tela antimosquitos. Es un efecto frecuente en visores de realidad virtual no suficientemente avanzados.

2.10. Sensores

Los sensores son los dispositivos encargados de recoger la información de los diferentes parámetros que controla el sistema de control centralizado como la temperatura ambiente, la existencia de un escape de agua o gas, la presencia de un intruso, etc., y enviársela a la central para que ejecute automáticamente las tareas programadas.

Los hay de diversos tipos: gas, temperatura, agua, humedad, luz, movimiento, rotura, etc., y están distribuidos por todo el domicilio, según la zona a vigilar/proteger son más adecuados unos sistemas que otros, y lo común suele ser utilizar una combinación de varios de ellos, cuantos más, mejor.

2.10.1. MPU-6050

El sensor InvenSense MPU-6050 contiene un acelerómetro MEMS y un giroscopio MEMS en un solo chip. Es muy preciso, ya que contiene hardware de conversión analógico a digital de 16 bits para cada canal. Por lo tanto, captura los canales x, y y z al mismo tiempo. El sensor utiliza el bus I2C para interactuar con el Arduino.

El MPU-6050 no es costoso, especialmente dado el hecho de que combina un acelerómetro y un giroscopio.

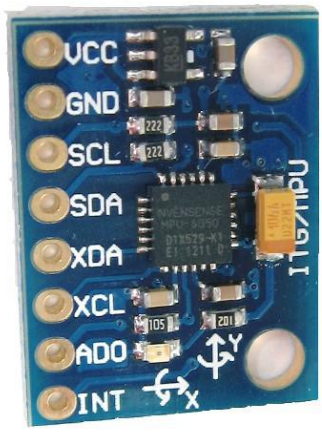


Figura 17: MPU-6050

Fuente: Elaboración Propia

CAPITULO III: MARCO APLICATIVO

3.1. Ingeniería de requisitos

Los requerimientos principales identificados de la aplicación se clasifican en funcionales y no funcionales.

3.1.1. Requisitos funcionales

√ EL usuario deberá poder iniciar la aplicación mediante un ejecutable o desde el propio entorno de desarrollo

√ El usuario deberá poder ver el menú principal al iniciar la aplicación

√ La aplicación debe contar con material multimedia

√ La aplicación deberá tener una interfaz gráfica de fácil lectura.

3.1.2. Requisitos no funcionales

√ La aplicación deberá ejecutarse en tiempo real con un retardo máximo de 2 mini-segundo de forma que no influya en la ejecución del misma.

√ El sistema dispondrá de una interfaz gráfica intuitiva e amigable.

√ El contenido del menú de la aplicación debe tener las opciones claras y ser conciso, de forma que no pueda llevar a confusión al jugador.

√ El acceso a la aplicación podrá ser llevado a cabo a través de un ejecutable o iniciando el proceso desde el propio entorno de desarrollo.

√ La aplicación funcionara en plataformas con Windows, Linux, Android y Web.

3.2. Aplicación de la Metodología SUM

3.2.1. Fase 1: Concepto

Para la elaboración de la aplicación se realizaron las investigaciones necesarias de aplicaciones y proyectos que aplican el mismo. De lo cual se obtuvo en el Marco Referencial toda la información para obtener un mejor resultado.

3.2.2. Fase 2: Planificación

Los casos de uso nos permitirán describir que hace el sistema desde el punto de vista del usuario, es decir, nos detallarán el uso del Sistema y cómo éste interactúa con los usuarios. De este modo se podrá observar las interacciones típicas en un usuario (actor) y el sistema, describiendo que se hace sin entrar en el cómo los hace.

El primer paso para escribir un caso de uso de forma eficiente es definir el conjunto de actores que podrán llevar las acciones a cabo. Un actor, es un elemento que se comunica con el sistema y que es externo al sistema en sí mismo, es decir la persona o sistema que utiliza el producto.

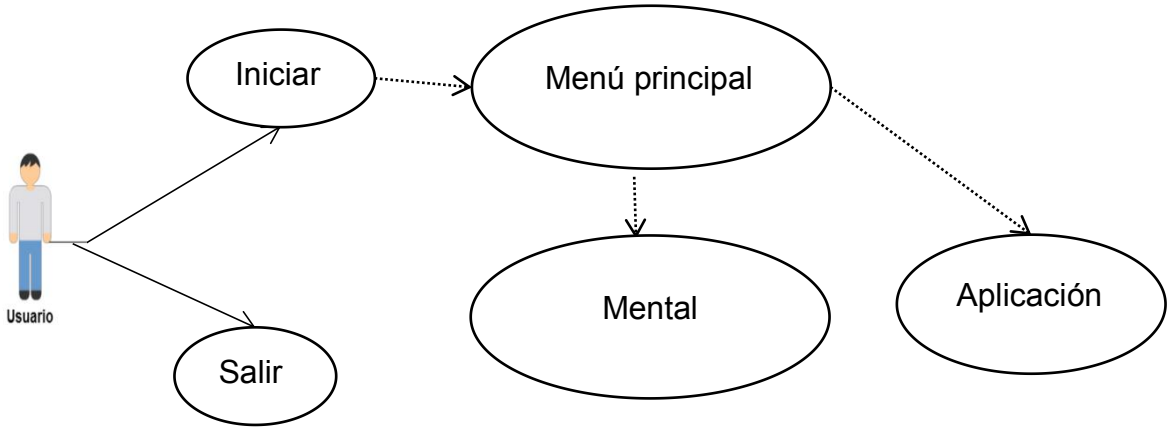


Figura 18: Diagrama de casos de uso

El siguiente paso será describir de forma detallada cada uno de estos casos de uso, de manera que se pueda ver cuál es la interacción entre el actor y el sistema, que condiciones deben darse para que el caso de uso pueda ser llevado a cabo o cuál es el flujo de interacción que da para realizar la acción.
En las siguientes tablas podremos ver las descripciones de todos los casos de uso mostrados en el diagrama de casos de uno mostrado anteriormente.

Identificador:	KV-01	Titulo:	Iniciar Aplicación
Actores:	Usuario		

Descripción:	El usuario inicia la aplicación
Precondiciones:	Dispones del ejecutable en le ordenador o celular Android
Flujo Normal:	1. Ejecutar el ejecutable de la aplicación 2. Espere a que cargue por completo la aplicación
Flujo Alternativo:	No aplica
Postcondiciones:	Se cargara la aplicación y se desplegara el menú principal

Tabla 1 KV-01: Iniciar la Aplicación

Identificador:	KV-02	Titulo:	Ver Menú Principal
Actores:	Usuario		
Descripción:	Permite al usuario ver el menú principal		
Precondiciones:	La aplicación debe estar iniciado		
Flujo Normal:	1. El usuario inicia la aplicación 2. La aplicación muestra las opciones dentro del menú 3. Para seleccionar una opción detener el cursor o touch en las opciones.		
Flujo Alternativo:	1. El usuario selecciona una opción del menú principal		
Postcondiciones:	Se muestra la selección del menú principal.		

Tabla 2 KV-02: Ver Menú principal

Identificador:	KV-03	Titulo:	Mental
Actores:	Usuario		
Descripción:	El usuario podrá usar la aplicación mental dentro de la aplicación implementados.		
Precondiciones:	La aplicación debe estar iniciado y seleccionar la opción de jugar.		
Flujo Normal:	1. Espere a que cargue por completo la aplicación		
Flujo Alternativo:	2. Volver al menú principal		
Postcondiciones:	Se ejecuta una acción dentro de la aplicación.		

Tabla 3 KV-03: Configuración

Identificador:	KV-04	Titulo:	Aplicación
Actores:	Usuario		
Descripcion:	El usuario podrá usar la aplicación Gamificada dentro de la aplicación implementados.		
Precondiciones:	La aplicación debe estar iniciado y seleccionar la opción de jugar.		
Flujo Normal:	1. Seleccionar un juego dentro del menú. 2. Comenzar la aplicación		
Flujo Alternativo:	Volver al menú principal		
Postcondiciones:	Se ejecutara acciones dentro de la aplicación.		

Tabla 4 KV-04: Aplicación

3.3. Diagrama de Actividades

Mediante el siguiente diagrama se expondrá de forma muy visual el funcionamiento general de la aplicación, desde el punto inicial hasta el punto final. En él se podrán ver las diferentes decisiones u opciones durante la ejecución del videojuego.

Al Arrancar el videojuego, se carga los recursos y se muestra el menú principal. En este punto el jugador tiene dos opciones, de comenzar el juego o ir a la opción mental.

Por último, si el jugador selecciona la opción de comenzar juego, este podrá acceder a una nueva partida.

Destacar que en cualquier momento de la ejecución del videojuego se podrá acceder al menú principal y se podrá abandonar el videojuego.

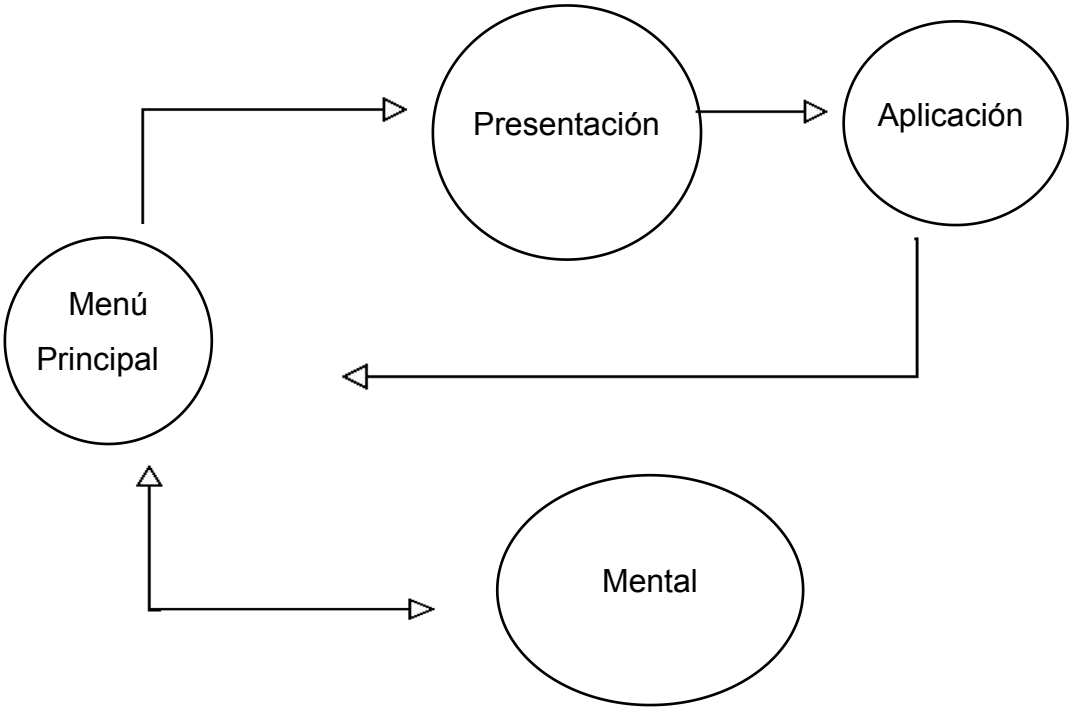


Figura 19: Diagrama navegacional

3.2.2. Fase 3: Elaboración

3.2.2.1. Escenas

Las escenas es una característica muy importante en Godot Engine, ya que contendrá todo el contenido de sus nodos y scripts correspondientes. Esto quiere decir que utilizar esta escena podrá tener acceso a todos los nodos hijo.

3.2.2.2. Comienzo del desarrollo

Para empezar a desarrollar se tendrá en cuenta las características que utiliza la plataforma Godot, como las relaciones que tienen las Escenas y sus nodos entre sí, donde se instancia las otras escenas creadas anteriormente.

Módulo 1: Menú Principal

La creación del menú principal se elaboró por medio de recursos 2D, para una mejor presentación, así mismo se implementaron distintas funciones, para darle

una animación adecuada al escenario como por ejemplo, el movimiento de traslación hacía, otro objeto, para lo cual se definieron los siguiente campos:

Principalmente contara con el entorno básico de un escenario, objetos, partículas de los escenarios, estructuras, etc... El escenario principal contara con las instancias realizadas para poder visualizar en el nodo principal, como las escenas que aparecerán son las del Jugador y otros.

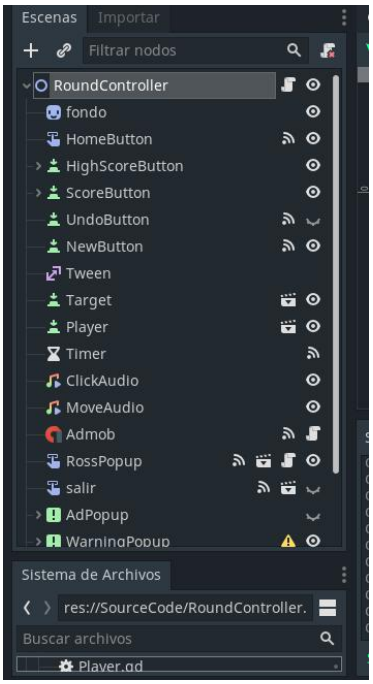


Figura 20: Nodos e instancias de la escena principal

Figura 21: Menú principal

Como se trata de una instancia, en estas escenas no se mostrará los nodos hijos, no preocuparse porque estos nodos estarán en la escena principal, donde ahí se Podrá editar sus parámetros y atributos correspondientes.

Como se puede ver la escena principal tiene el nombre RoundController, donde el tipo de nodo es Node y esto nos permitirá poder instanciar todo tipo de nodos hijos, porque es el nodo estándar y padre de todos los anteriores.

Módulo 2: Mental

La repetición es el sistema de memorizar más usado por los estudiantes, pero no se utiliza normalmente de la manera más efectiva posible. En este módulo consiste en poder aprender técnicas de memorización como el casillero mental y otros parecidos, que es una construcción memorística que nos permitirá almacenar datos en nuestra memoria de forma ordenada.

Figura 22: Casillero Mental

Te permitirá la construcción de tu casillero mental, ya que es completamente necesario para cualquier estudiante. Es más, puedo afirmar sin temor a equivocarme que nunca será un eficaz estudiante aquel que no tenga ni sepa usar

un buen casillero mental.

Módulo 3: Cálculo Mental

Consiste en poder armar números que se suma o multiplican entre sí, de forma aleatoria.

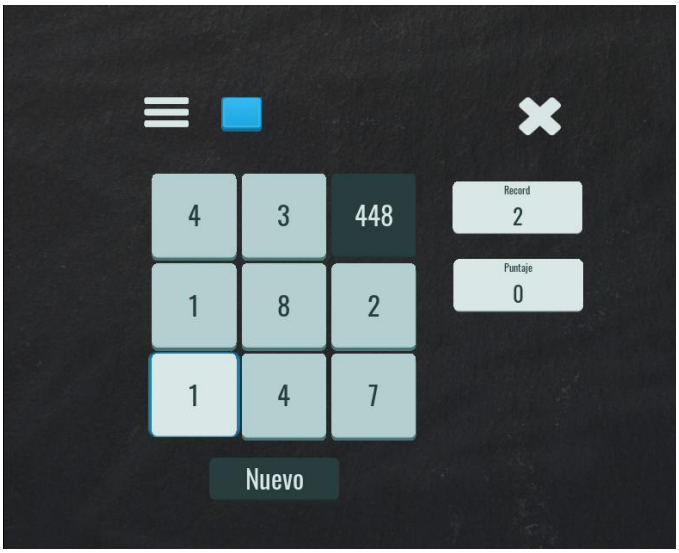


Figura 23: Cálculo mental en ejecución

El juego esta que debes hallar el resultado que se genera de forma aleatoria, podemos ir un derecha, izquierda, arriba o abajo hasta llegar al resultado.

Cada vez que obtengas un resultado correcto sumaras puntos. Si el pontaje es superior a tu Record este aumentara, guardandose en base de datos json.

Podemos presionar el botón nuevo para volver a intentar cuantas veces sea necesario.

Al iniciar el juego te aparecerá un mensaje creado con el plugin AdMob, para poder monetizar la aplicación, lo podemos cambiar por cualquier publicidad.

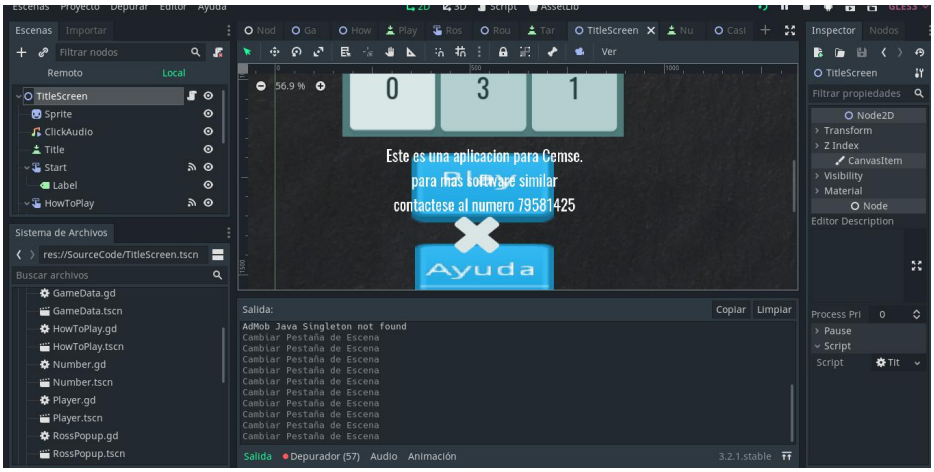


Figura 24: Mensaje emergente con AdMob

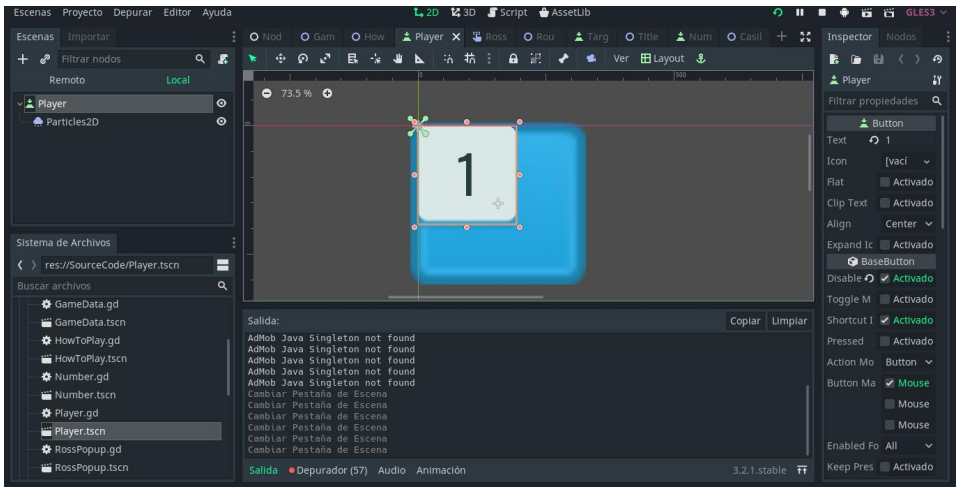


Figura 25: Nodo jugador

El jugador cuenta con partículas 2d, animaciones, etc. para ser más agradable en la experiencia de juego.

3.2.3. Fase 4: Beta

Al completar todas las iteraciones dentro de la fase de elaboración se procede a liberar la primera versión de la aplicación.

3.2.4. Fase 5: Cierre

Una vez completada la versión beta se proceda al poner a disposición de distintos

usuarios para la evaluación del desarrollo de la aplicación.

Al obtener una evaluación satisfactoria de la versión beta del videojuego podemos cerrar el proyecto como primera versión.

3.2. Aplicación de Framework D6

3.2.1. Definir los Objetivos de Negocio:

- √ Borrar el mito de que la matemática es una actividad especializada no útil para la vida diaria.
- √ Motivar a los estudiantes de cursos de matemáticas a que estudien y practiquen los conocimientos adquiridos en las clases.
- √ Fomentar el autoaprendizaje para minimizar la dependencia de los conceptos transmitidos en el salón de clases.

3.2.2. Delinear los Comportamientos Objetivo

Se espera que los jugadores utilicen este sistema para resolver más problemas de matemáticas respecto a los resueltos durante el desarrollo de un curso de matemática.

3.2.3. Describir a los Jugadores

Para este modelo de Gamificación, los jugadores serán descritos de dos formas: demográficamente y por la clasificación de jugadores descrita por Richard Bartle

3.2.4. Determinar los Ciclos de Actividad

3.2.4.1. Ciclos de Progreso

Cuando el jugador completa un problema correctamente, recibe puntos de experiencia. A medida que el usuario acumula estos puntos y cuando llega a ciertos umbrales predeterminados, el jugador aumenta su nivel. Esto refuerza al jugador en el hecho que ha adquirido experiencia desde que ha empezado a utilizar el sistema.

3.2.4.2. Ciclo de Compromiso

El sistema está basado en la solución de problemas de matemática, así que el ciclo de actividad principal se centra en esto.

Una vez solucionado el problema, se desbloquean nuevos problemas para que el

usuario los resuelva, repitiendo el ciclo de planteamiento, solución y retroalimentación.

3.2.5. ¿Dónde se Encuentra la Diversión?

Para determinar cómo el jugador encuentra la diversión dentro del sistema, se plantea cuatro aspectos clave que debe tener un juego para hacer que sus jugadores sientan apego y se sientan comprometidos emocionalmente con este (Lazzaro, 2002). Es importante que un juego que quiera tener una audiencia diversa, explote más de uno de los aspectos. Este modelo utiliza tres de ellos para describir cómo el jugador se va a divertir utilizándolo.

3.2.5.1. Diversión Difícil

La diversión difícil se refiere a la diversión que surge cuando a una persona se le presenta un reto y disfruta resolviéndolo. Los retos hacen que los jugadores se enfoquen en la actividad que realizan, elijan una estrategia de solución y recompensan el progreso, haciendo que el jugador tenga una sensación de triunfo personal. Los retos deben ser balanceados mediante niveles de dificultad y progreso para mantener un buen nivel de frustración y recompensa.

Este aspecto es el que predomina en este modelo. Se espera que los jugadores se sientan comprometidos con solucionar los retos que se les presentan. Los problemas de programación son problemas concretos que muy a menudo tienen más de una estrategia de solución posible, con lo cual el jugador puede explotar su creatividad y su capacidad de solución de problemas, a la vez que se intenta convertir la frustración de encontrar un problema difícil en un reto que proporcione satisfacción cuando se logre solucionar.

3.2.5.2. El Factor Social

Este aspecto se refiere al disfrute que obtienen los jugadores cuando interactúan con otras personas dentro o por fuera del juego. El factor social crea emociones y vínculos tan importantes que algunas personas incluso juegan juegos que no les gustan simplemente para pasar tiempo con su amigo. La rivalidad, el trabajo en equipo y los objetivos en común son estrategias de los juegos para explotar estas emociones.

El factor social está presente en este modelo mediante tres aspectos principales. Para explotar el altruismo y las sensaciones de bienestar generadas al ayudar a los demás, cuando un jugador logra solucionar un problema podrá escribir consejos para otros jugadores que no lo hayan resuelto aún. Adicionalmente, el jugador podrá compartir sus logros, los problemas resueltos y otros aspectos de su experiencia. Por último, diferentes jugadores pueden unirse para resolver retos de alta dificultad que requieran las habilidades de personas que hayan desbloqueado distintos tipos de jugadores.

3.2.6. Desplegar las Herramientas Adecuadas.

3.2.6.1. Triada PBL (conocida por sus siglas en inglés: Points, Leaderboards, *Badges)*

Aunque no es la mecánica central de este modelo, estos tres elementos siguen siendo útiles para recompensar a los jugadores cuando alcanzan un objetivo dentro del sistema. Los puntos de experiencia funcionan como un valor cuantitativo que permite al jugador saber qué tanto ha avanzado desde el punto cuando empezó a utilizar el sistema.

Los tableros de puntaje servirán para motivar a los jugadores más competitivos, mostrando por cada problema cuáles son los mejores tiempos de solución, quiénes son los usuarios con más problemas terminados, etc.

3.3. ANALISIS DE CALIDAD, SEGURIDAD Y COSTOS

3.3.1. Calidad

Existen muchos autores de definen la Calidad del Software, Pressman define a la calidad del Software como “la concordancia con los requisitos funcionales y de rendimientos explícitamente establecidos, estándares de desarrollo explícitamente documentados y características implícitas que se espera de todo software desarrollado profesionalmente”.

Dicho esto diremos que la Calidad del Software comprende distintos aspectos como estética (que sea agradable a la vista), funcionalidad (que sea fácil de usar), eficiencia (que se ejecute con rapidez y precisión de procesos), etc.

Las características de calidad que se tomaron en cuenta para evaluar las

propiedades de esta aplicación móvil se determinan en el modelo de la NORMA ISO/IEC 9126.

3.3.1.1. Técnica ISO 9126

El objetivo principal de esta técnica es alcanzar la calidad necesaria para satisfacer las necesidades del cliente. La calidad según esta norma ISO 9126 puede ser medida de acuerdo a los factores:

- √ Usabilidad
- √ Funcionalidad
- √ Confiabilidad
- √ Mantenibilidad
- √ Portabilidad

3.3.1.1.1. Usabilidad

La usabilidad consiste de un conjunto de atributos que permite evaluar el esfuerzo necesario que deberá invertir el usuario para utilizar el sistema, es decir realizar una serie de preguntas que permiten ver cuán sencillo, fácil de aprender y manejar es para los usuarios. Esta comprensión por parte de los usuarios con relación al sistema evalúa los siguientes pasos:

- √ Entendimiento
- √ Aprendizaje
- √ Operatividad
- √ Atracción
- √ Conformidad de uso

En la siguiente tabla se observa estos criterios en niveles de porcentajes a los que llego el sistema en cuanto a su comprensibilidad, para el usuario, y posteriormente se da el porcentaje final de usabilidad del sistema. Se realizó una encuesta al usuario final sobre el manejo, la compresión y la facilidad de aprender el sistema para medir la usabilidad según la siguiente tabla.

PRECUNTAS	RESPUESTAS	PORCENTAJE

	SI B(1-10)	NO(1-10)	
¿El acceso al Sistema es complicado?	0	10	100%
¿Las respuestas del sistema son de su comprensibilidad?	9	1	100%
¿Tiene alguna dificultad en realizar los procesos del sistema?	2	8	80%
¿La interfaz de la aplicación es amigable y entendible a su parecer?	9	1	90%
¿Son comprensibles y satisfactorios los datos que se muestran en el sistema?	10	0	100%
¿El sistema coadyuva de manera eficiente en los procesos que se realizaban anteriormente?	9	1	90%
¿El sistema satisface los requerimientos de la Constitución?	8	2	80%
¿El sistema es de fácil uso bajo su criterio?	10	0	100%
¿Es aplicable el sistema para el aprendizaje de sus estudiantes?	8	2	80%
PROMEDIO TOTAL			91.1%

Tabla 5 KV-05: Encuesta de usabilidad del sistema [Fuente: Elaboración propia]

Usabilidad = 91%

De acuerdo a los datos obtenidos en la tabla de usabilidad, se concluye que el sistema tiene una usabilidad del 91%. Esto indica que el usuario de la aplicación tiene una conformidad hacia el sistema y que este podrá coadyuvar de alguna manera a solucionar los problemas de la institución planteados anteriormente.

3.3.1.1.1. Funcionalidad

La funcionalidad examina si el sistema satisface los requisitos funcionales esperados. El objetivo es revelar problemas y errores en lo que concierne a la funcionalidad del sistema y su conformidad al comportamiento, expresado o deseado por el usuario. Se cuantifica el tamaño y la complejidad del sistema en términos de las funciones de usuario, puede ser valorado mediante el Punto Función. Se basa en la contabilización de cinco parámetros los cuales se desarrollan a continuación:

Número de entradas de usuario: Se refiere a cada entrada de control del usuario que proporciona diferentes datos al sistema.

Para calcular el Punto Función se utilizará la siguiente fórmula:

$$Punto\ función\ (PF) = Total * (X + Min * Sum(F_i))$$

Donde:

PF: Medida de la funcionalidad.

X: Confiabilidad del proyecto, varía entre el 1% a 100%

Min (y): Error mínimo aceptable al de la complejidad, el margen de error es igual a 0.01.

***F_i*:** Son los valores de ajuste de la complejidad, donde i=1 a i=14.

En la siguiente tabla se calcula el punto función, los cuales miden el software desde una perspectiva del usuario, dejando de lado los detalles de codificación.

PARAMETROS	CANTIDAD	PESO SIMPLE	PESO MEDIO	PESO COMPLEJO	FACTOR DE PESO
E.I. Entrada externa	150	3	4	6	600
E.O. Salida	150	4	5	7	1050

externa					
E.Q. Consulta externa	1	3	4	6	3
I.L.F. Archivo lógico interno	1	7	10	15	7
E.I.F. Archivo de interfaz externo.	1	5	7	10	10
TOTAL					1670

Tabla 6 KV-06: Conteo de parámetros PF
[Fuente: Informática Vol. IV, Garzón y Ángeles, 2003]

Entrada externa (EI -> External input). Pantallas donde el usuario ingresa datos

Salida externa (EO -> External output). Informes, gráficos, listado de datos

Consultas externa (EQ -> External query). Recuperar y mostrar datos al usuario(Buscar)

Archivo logico interno (ILF -> Internal logical File).

Archivo de interfaz externo (EIF -> External Interface File).

IMPORTANCIA	0	1	2	3	4	5
ESCALAS	SIN IMPORTANCIA	INCREMENTAL	MODERADO	MEDIO	SIGNIFICATIVO	ESENCIAL
1.- ¿Requiere el sistema copias de seguridad y de recuperación fiable?	X					
2.- ¿Se requiere comunicación de datos especializadas para transferir información a la					X	

aplicación u obtenerlas de ellas?						
3. ¿Existe funciones de procesos distribuidos?	X					
4. ¿Es critico el rendimiento?						X
5.- ¿Será ejecutado el sistema en un entorno existente y fuertemente utilizado?					X	
6.- ¿Requiere el sistema entrada de datos interactiva?					X	
7.- ¿Requiere la entrada de datos interactiva que las transacciones de entrada se lleven a cabo sobre múltiples pantallas o variadas operaciones?				X		
8.- ¿Se actualiza los archivos de forma interactiva?			X			
9.- ¿Son complejas las entradas, salidas, los archivos o las peticiones?			X			
10.- ¿Es complejo el procesamiento interno del sistema?					X	
11.- ¿Se ha diseñado el código para ser reutilizado?						X
12.- ¿Están incluidas en el diseño la conversión y la instalación?				X		
13.- ¿Se ha diseñado el sistema para soportar múltiples						X

instalaciones en diferentes instituciones?						
14.- ¿Se ha diseñado la aplicación para facilitar los cambios y para ser fácilmente utilizados por el usuario?						X
CUENTA TOTAL	$\sum fi = 46$					

Tabla 7 KV-07: Ajuste de complejidad [Fuente: Informática Vol. IV, Garzón y Ángeles, (2003)]

Para el ajuste se utiliza la ecuación:

PF = Cuenta Total *(Grado de confiabilidad + Tasas de Error * Σ(fi))

Punto Función (PF) = 1670 + (0,65 + 0,01 * 46)

Punto Función Resultante (PF)=1671,11

Para hallar el punto función ideal al 100% de los factores seria 70:

PF=1981 + (0,65 + 0,01 * 70)

PF IDEAL =1982,35

Calculando del % de funcionalidad real:

PF Real = (1671,11 /1982,35**)*100=84,23**

Por tanto:

Funcionalidad = 84%

Interpretando, la aplicación tiene una funcionalidad o utilidad del **84**% para la institución, lo que indica que él es sistema cumple con los requisitos funcionales de forma satisfactoria.

3.3.1.1.2. Confiabilidad

La confiabilidad permite evaluar la relación entre el nivel de funcionalidad y la cantidad de recursos usados, es decir, representa el tiempo que el software está disponible para su uso, la misma se calcula utilizando la privacidad de que un sistema presente fallas:

Comportamiento con respecto al tiempo: Atributos de software relativos a los tiempos de respuesta y de procesamiento de los datos.

Comportamiento con respecto a Recursos: Atributos software relativo a la cantidad de recursos usados y la duración de su uso en la realización de funciones.

La función a continuación muestra el nivel de confiabilidad del sistema:

$$F(t) = (\text{Funcionalidad}) * e^{(-\lambda t)}$$

Se ve el trabajo hasta que se observa un fallo en un instante t, la función es la siguiente:

Probabilidad de hallar una falla: $P(T \leq t) = F(t)$

Probabilidad de no hallar una falla: $P(T > t) = 1 - F(t)$

Donde:

Funcionalidad = 0,84

$\lambda = 0.14$ (1 error cada 7 ejecuciones)

Tomemos un tiempo t de 12 meses

Ahora hallando la confiabilidad del sistema:

$$F(12) = (0,84) * e^{(-0.14*12)}$$

$$F(12) = 0,156$$

Remplazando en la fórmula de no hallar una falla se tiene:

$$P(T > t) = 1 - F(t)$$

$$P(T > t) = 1 - 0,156$$

$$P(T > t) = 0,84$$

Con este resultado podemos decir que la probabilidad que el sistema no presente fallas es de 0,84 y que presente fallas es del 0,16.

Confiabilidad = 84%

Con este resultado se concluye que la aplicación tiene un grado de confiabilidad del 82% durante los próximos 12 meses.

3.3.1.1.3. Mantenibilidad

La Mantenibilidad se refiere a los atributos que permiten medir el esfuerzo necesario para realizar modificaciones al software, ya sea por la corrección de errores o por el incremento de funcionalidad. Para hallar Mantenibilidad del sistema se utiliza el índice de madurez de software (IMS), que proporciona una indicación de la estabilidad de un producto de software.

Para calcular el índice hacen falta una serie de medidas anteriores:

Mt = número de módulos en la versión actual.

Fm = número de módulos en la versión actual que han sido modificados.

Fa = número de módulos en la versión actual que han sido añadidos.

Fe = número de módulos de la versión anterior que se han eliminado en la versión actual.

A partir de estas, el IMS se calcula de la siguiente forma:

$$IMS = \frac{[Mt - (Fa + Fm + Fe)]}{Mt}$$

$$IMS = \frac{[2 - (0 + 1 + 0)]}{2}$$

IMS = 0.5

La interpretación a este resultado establece un 50%, lo que indica que no requiere de mantenimiento inmediatamente.

Mantenibilidad = 50%

3.3.1.1.4. Portabilidad

Capacidad del producto o componente de ser transferido de forma efectiva y eficiente de un entorno hardware, software, operacional o de utilización a otro.

Esta característica se subdivide a su vez en las siguientes sub características:

Adaptabilidad. Capacidad del producto que le permite ser adaptado de forma efectiva y eficiente a diferentes entornos determinados de hardware, software, operacionales o de uso.

Capacidad para ser instalado. Facilidad con la que el producto se puede instalar y/o desinstalar de forma exitosa en un determinado entorno.

Capacidad para ser reemplazado. Capacidad del producto para ser utilizado en lugar de otro producto software determinado con el mismo propósito y en el mismo entorno.

Para el cálculo de la portabilidad se tomó en cuenta la siguiente tabla que contiene las características que se mencionaron anteriormente.

FACTOR DE PORTABILIDAD	VALOR %
Puede ser transferido de un entorno a otro	90
Se puede adaptar a otros ambientes con facilidad (Instituciones similares)	100
Es fácil de Instalar	100
Es capaz de reemplazar a una aplicación similar	90
TOTAL	95%

Tabla 8 KV-08: Portabilidad
[Fuente: Elaboración propia]

La interpretación a este resultado significa que nuestra aplicación móvil tiene la capacidad de ejecutarse en diferentes dispositivos.

Portabilidad = 95%

3.3.2. Seguridad

La seguridad es la capacidad de protección de la información y los datos de manera que las personas no autorizadas puedan leerlas y/o modificarlas. Esta

característica se subdivide a su vez en las siguientes sub características:

3.3.2.1. Seguridad a nivel sistema operativo

En el presente proyecto se aplicó la seguridad todos los parámetros de seguridad al momento de exportar para una plataforma en concreto.

3.3.2.2. Seguridad a nivel Aplicativo Móvil

El aplicativo móvil podrá ser controlado por un administrador.

Se desarrolló una conexión a Firebase para inicio de sesión de usuarios para la restricción del acceso a usuario no autorizado. Este verifica y autoriza el ingreso al sistema a los usuarios por medio de usuario y contraseña donde estos están encriptados por un cifrado que da Firebase.

3.3.3. Costos

Para determinar el Costo Total del proyecto se tomara en cuenta el costo de software, costo de implementación de la aplicación y elaboración del proyecto.

3.3.3.1. Costo de la aplicación

Para determinar el costo de la aplicación se usa el modelo COCOMO II orientado en los puntos de función, para calcular el Esfuerzo, necesitaremos hallar la variable KDLC (Kilolíneas de código), donde los PF son 307,2 y las líneas por cada PF equivalen a 32 según vemos en la tabla que se ilustra a continuación:

LOC por punto de función			
Lenguaje	**LOC/FP**	**Lenguaje**	**LOC/FP**
Ensamblador	320	Basic ANSI/Quick/Turbo	64
Macroensamblador	213	Java	53
C	150	Visual C++	34
Fortran	106	Foxpro 2,5	34
Cobol	106	Visual Basic	32
Pascal	91	Delphi	29
Cobol ANSI 85	91	C++	29
Basic	91	Visual Cobol	20
RPG	80	Clipper	19
PL/I	80	Power Builder	16
Ada	71	Hoja de Calculo	6

Tabla 9 KV-09: Conversión de Puntos de Función
[Fuente: QSM, 2018]

Así pues tras saber que son 32 LDC por cada PF, el resultado de los KDLC será el siguiente:

$$KLDC = \frac{(PF * lineasdecodigoporcadaPF)}{1000}$$

$$KLDC = \frac{(307{,}2 * 32)}{1000}$$

KLDC = 9,8304

Así pues, en nuestro caso el tipo semi - acoplado será el más apropiado ya que el número de líneas de código es menor a los 5000, y además el proyecto tiene una complejidad media, por consiguiente, los coeficientes que usaremos serán las siguientes:

PROYECTO SOFTWARE	**a**	**E**	**c**	**d**
ORGANICO	3,2	1,05	2,5	0,38
SEMI-ACOPLADO	3,0	1,12	2,5	0,35
EMPOTRADO	2,8	1,20	2,5	0,32

Tabla 10 KV-10: Modelo COCOMO Fuente: [Pressman R., 1999]

CONDUCTORES DE COSTE	VALORACIÓN					
	Muy bajo	*Bajo*	*Nominal*	*Alto*	*Muy alto*	*Extr. alto*
Fiabilidad requerida del software	0,75	0,88	1.00	**1,15**	1,40	-
Tamaño de la base de datos	-	0,94	**1.00**	1,08	1,16	-
Complejidad del producto	0,70	**0,85**	1.00	1,15	1,30	1,65
Restricciones del tiempo de ejecución	-	-	1.00	**1,11**	1,30	1,66
Restricciones del almacenamiento principal	-	-	**1.00**	1,06	1,21	1,56
Volatilidad de la máquina virtual	-	0,87	**1.00**	1,15	1,30	-
Tiempo de respuesta del ordenador	-	0,87	1.00	**1,07**	1,15	-
Capacidad del analista	1,46	1,19	1.00	**0,86**	0,71	-
Experiencia en la aplicación	1,29	1,13	1.00	0,91	**0,82**	-
Capacidad de los programadores	1,42	1,17	1.00	0,86	**0,70**	-
Experiencia en S.O. utilizado	1,21	1,10	**1.00**	0,90	-	-
Experiencia en el lenguaje de programación	1,14	1,07	1.00	**0,95**	-	-
Prácticas de programación modernas	1,24	1,10	**1.00**	0,91	0,82	-
Utilización de herramientas software	1,24	1,10	1.00	**0,91**	0,83	-
Limitaciones de planificación del proyecto	1,23	**1,08**	1.00	1,04	1,10	-

Tabla 11 KV-11: Conductores de costo [Fuente: Calculo de costos de producción de software, Castillo, 2017]

FAE=1,15*0,85*1,11*1*1*1,07*0,86*0,82*0,70*1*0,95*1*0,91*1,08

FAE= 0,535084806

Justificación:

Atributos de software

Fiabilidad requerida del software: Si se produce un fallo por la venta de un producto, o fallo en algún registro del costo del producto, puede ocasionar grandes pérdidas a la empresa (Valoración Alta).

Tamaño de la base de datos: La base de datos de nuestra aplicación móvil será muy bajo.

Complejidad del producto: La aplicación no va a realizar cálculos complejos (Valoración Baja).

Atributos de hardware

Restricciones del tiempo de ejecución: En los requerimientos se exige alto rendimiento

(Valoración Alta).

Restricciones del almacenamiento principal: No existen restricciones al respecto (Valoración Nominal).

Volatilidad de la máquina virtual: Se usarán dispositivos móviles con sistemas operativos Android (Valoración Nominal).

Tiempo de respuesta del ordenador: Deberá ser amigable con el usuario (Valoración Alta).

Atributos del personal

Capacidad del analista: Capacidad alta relativamente, debido a la experiencia en análisis en proyecto similar (Valoración Alta)

Experiencia en la aplicación: Se tiene cierta experiencia en aplicaciones de esta envergadura (Valoración muy alta).

Capacidad de los programadores: Teóricamente deberá tenerse una capacidad muy alta por la experiencia en anteriores proyectos similares (Valoración muy alta).

Experiencia en S.O. utilizado: Con Android la experiencia es a nivel usuario (Valoración Nominal).

Experiencia en el lenguaje de programación: Es relativamente alta, dado que se controlan las nociones básicas y las propias del proyecto (Valoración Alta).

Atributos del proyecto

Prácticas de programación modernas: Se usarán prácticas de programación mayormente convencional (Valoración Nominal).

Utilización de herramientas software: Se usarán herramientas estándar que no exigirán apenas formación, de las cuales se tiene cierta experiencia (Valoración Alta).

Limitaciones de planificación del proyecto: Existen pocos límites de planificación.

(Valoración Baja).

Cálculo del esfuerzo del desarrollo:

E = a KLDC e * FAE = 2,4 * (16,089) ^1,05 * 0,535084806 = 23,74 [pers/mes]

Cálculo tiempo de desarrollo:

T = c Esfuerzo d = 2,5 * (23,74) ^0,35 = 7.57 [meses]

Personal promedio:

P = E/T

P=23,74 / 7,57 = 3,1 personas

Según este resultado será necesario un equipo de 3 personas trabajando alrededor de 9 meses.

Ahora pues sabemos que el salario promedio de un programador oscila entre 650 Bs. y 950 Bs, estimando con el costo mínimo tomaremos 400Bs para calcular el costo de la aplicación.

Costo de la aplicación = Numero de programadores*Salario de un programador

Costo de la aplicación = 3*500

Costo de la aplicación = 1200 Bs.

3.3.3.2. Costo de software desarrollado

Debido a que la aplicación está desarrollada con software libre y que la misma es

utilizada por diferentes empresas será de libre distribución.

3.3.3.3. Costo de Elaboración del Proyecto

Por otra parte el desarrollo como tal del proyecto comprende la siguiente serie de gastos que se detallan a continuación:

DETALLE	IMPORTE(Bs)
Análisis y diseño de la aplicación	500
Material de escritorio	100
Investigación en Internet	300
Bibliografía	100
Otros	200
TOTAL	**1200**

Tabla 12 KV-12: Costo Elaboración del Proyecto [Fuente: Elaboración propia]

3.4. Construcción Casco realidad virtual con Arduino y Godot Engine

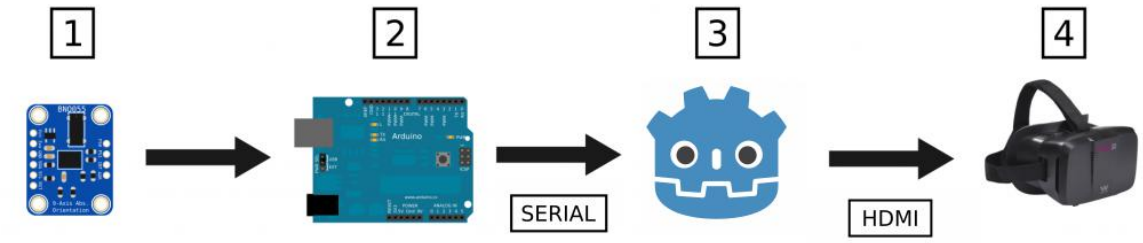


Figura 26. Funcionamiento del casco [Fuente: Elaboración propia]

√ El MPU6050 se coloca en el casco para calcular su inclinación y así saber hacia donde mira el usuario. El MPU6050 envía las lecturas de rotación a la placa Arduino.

√ La placa Arduino junta el valor de los tres ángulos XYZ en un solo mensaje que envía por Serial a un ordenador con Godot Engine.

√ Godot, mediante un script Python y la librería PySerial, decodifica el mensaje que ha enviado Arduino, extrae los ángulos X, Y, Z y rota la cámara de la

escena para que el usuario pueda ver el entorno.

√ El ordenador envía la imagen de la escena 3D a través de HDMI a una pantalla de 5'' colocada dentro del casco.

Al mover la cabeza variará la lectura de la IMU, que a su vez modificará la rotación de la cámara de Godot. Esto producirá una sensación de Realidad Virtual.

3.4.1. Programa Arduino y el MPU6050

Para que el usuario tenga la sensación de estar dentro de la escena virtual, la cámara de la escena 3D debe moverse hacia donde la persona esté mirando. Si el usuario gira la cabeza hacia la derecha, la cámara también debe moverse en esta dirección y así poder ver los objetos que hay al lado. Por lo tanto el primer problema al que nos enfrentamos es saber en qué dirección está mirando el usuario, y para ello hay que saber la rotación de su cabeza.

Una MPU es una combinación de Giroscopio, Acelerómetro y algunas veces Magnetómetro que permite medir inclinaciones. El BNO055 es una IMU de 9 grados de libertad (9 Degrees of Fredoom), lo que quiere decir que podrá medir la rotación alrededor de los 3 ejes X, Y, Z. Si enganchamos la IMU al casco de RV, su rotación será la misma que la cabeza del usuario.

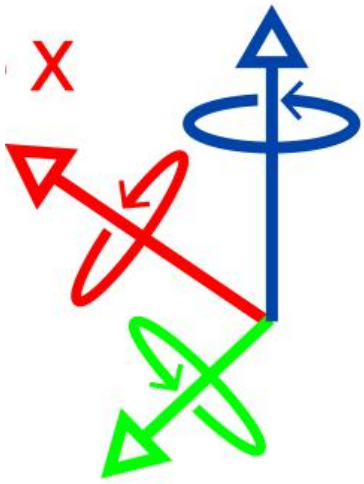


Figura 27. Dirección de los ejes [Fuente: Elaboración propia]

3.4.2. Instalar librerías

Para conseguir lecturas precisas de una MPU, normalmente se aplican algoritmos matemáticamente complicados como el Filtro de Kálman o el Filtro de Madgwick.

Por suerte el MPU dispone de librerías que nos ahorran el trabajo duro y nos dan las lecturas de los ángulos ya preparadas: la librería AdafruitSensor y la librería Adafruit BNO055. Una vez las tengas descargadas, abre el IDE de Arduino y ve a la pestaña Sketch→Importar Librería→Añadir librería y busca los dos ficheros .zip que acabas de descargar.

Puede que te salte un error a la hora de importarlas, diciendo que la librería contiene carácteres erróneos. Si es así, elimina los carácteres “-” y “_” del nombre de los archivos .zip.

3.4.3. Conexiones

Las conexiones entre Arduino y el BNO055 serán:

Arduino 5V → IMU VCC

Arduino GND → IMU GND

Arduino SDA → IMU SDA

Arduino SCL → IMU SCL

Normalmente los pines SDA y SCL de Arduino son los pines A4 y A5 respectivamente. Sin embargo, deberías leerte esta referencia antes de conectar nada.

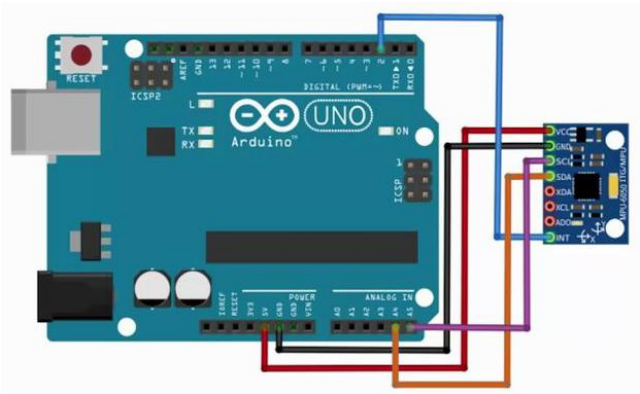


Figura 28. Conexión [Fuente: Elaboración propia]

3.4.4. Código de Arduino

Vamos a escribir un código que lea los tres ángulos de la IMU, los junte en un único string y lo envíe por serial. Todos los mensajes enviados tendrán este formato:

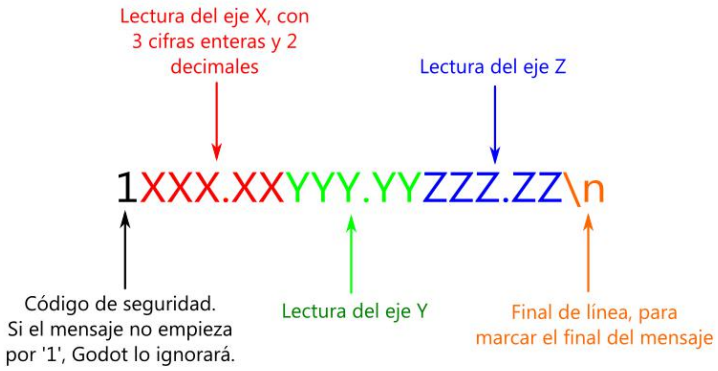


Figura 29: Formato del código [Fuente: Elaboración propia]

Este será el código que tendrás que que se cargada en la placa Arduino. Se debe tener en cuenta que quizá se tendrá que modificar las correcciones de los ángulos dependiendo del modelo exacto de tu IMU y de como se coloque sobre el casco de RV.

3.4.4. Configurando Godot Engine

Figura 30: Ventana del juego [Fuente: Elaboración propia]

Los cascos de RV están construidos de tal forma que cada ojo sólo puede ver la mitad de la pantalla que haya dentro. Por tanto, si se renderiza la misma imagen a cada mitad de la pantalla (con una pequeña variación de ángulo) se conseguirá engañar al usuario y crear la sensación de 3D.

Godot tiene una cámara especial para hacer Realidad Virtual que ya viene con visión estereoscópica, pero está pensada para las plataformas de RV profesionales y nos daría algunos problemas para nuestro sistema casero. Esto nos obliga a crear nuestra propia cámara estereoscópica desde cero.

La forma de hacerlo será creando dos Viewports distintos, cada uno de los cuáles tendrá como hijo un Spatial (que actuará de “cabeza”). A su vez, estos Spatials serán padres de una cámara (los “ojos” de la cabeza).

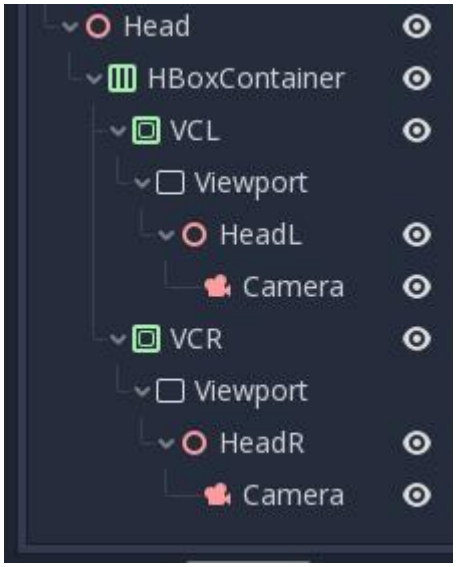


Figura 31: Configuración de nodos [Fuente: Elaboración propia]

El script que se programo más adelante se encargará de girar estas dos cabezas (“HeadR” y “HeadL”) para igualar el ángulo que detecta la IMU del casco.

Debe haber dos cámaras, por dos razones: la primera es que un nodo Viewport sólo cogerá la imagen de una cámara que tenga como hija, por tanto cada cámara tiene que estar emparentada a su Viewport.

La segunda razón es que los Viewports no son objetos que tengan una posición en el mundo 3D. Por tanto, si ponemos un Spatial como padre de los Viewports, la rotación de este Spatial padre no se transmitirá a los Viewports, ni a ninguno de los nodos hijos de los Viewports. A todos los efectos es como si las cámaras no estuvieran emparentadas a ninguno de los nodos que queden por encima de los Viewports.

3.4.5. Configuración Python en Godot Engine

Extensión que permite programar scripts de Godot en lenguaje Python. Esto es muy útil porque Python dispone de librerías como PySerial pensadas para poder leer el puerto de serie fácilmente.

Para poder utilizar Python con Godot, debemos instalar la librería PythonScript.

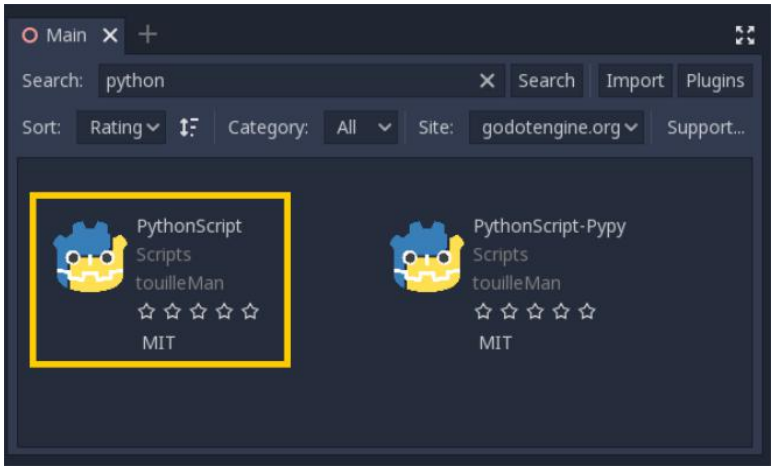


Figura 32: Libreria PythonScript [Fuente: Elaboración propia]

Godot-Python viene con las librerías básicas de Python, pero necesitamos comunicación serial para la cual instalamos PySerial .

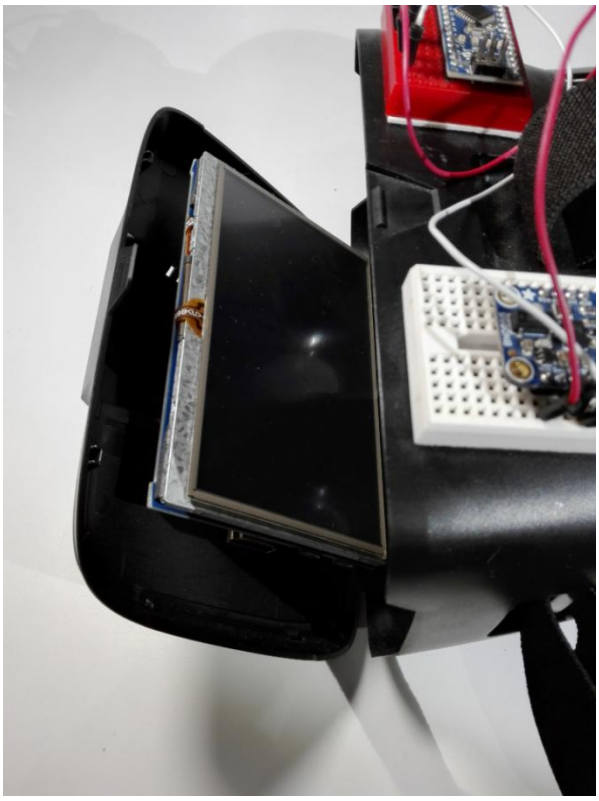


Figura 33: Casco con pantalla [Fuente: Elaboración propia]

CAPITULO IV: CONCLUSIONES Y RECOMENDACIONES

4.1. Conclusiones

El desarrollo de aplicaciones orientadas al aprendizaje con la tecnología de videojuegos genera un gran aporte a la educación, debido a que se puede mostrar de una forma dinámica las técnicas como la Gamificación, aprendizaje y memoria.

El objetivo general fue cumplido de manera aceptable ya que se desarrolló un prototipo basado en técnicas de aprendizaje, como la Gamificación para motivar al aprendizaje de la matemática a los estudiantes de sexto de secundaria, además de emplear la metodología de desarrollo SUM la cual es ideal para el desarrollo de aplicaciones tipo videojuego.

- Se logró diseñar una interfaz amigable y de fácil manipulación para el usuario.
- Se logró implementar técnicas adecuadas para motivar a los estudiantes de sexto de secundaria.
- Se logró implementar técnicas para desarrollar las capacidades mentales de los estudiantes de sexto de secundaria.
- Se logró diseñar un prototipo Hardware Libre que pueda interactuar con la aplicación.
- Se logró realizar la aplicación multiplataforma tanto para Windows, Linux y Android.
- Se logró realizar el manual de usuario correspondiente a la aplicación.

4.2. Recomendaciones

A partir del presente trabajo se propone las siguientes recomendaciones, con el fin de buscar el mejoramiento del sistema.

- Se recomienda para trabajos futuros una aplicación web para mejorar el acceso de los estudiantes.
- Convendría utilizar una metodología con más tiempos cortos para juegos serios.
- Se recomienda realizar un módulo para el control de los usuarios.
- Se recomienda realizar el prototipo de Hardware Libre más interactivo con los

juegos de nuestra aplicación.

BIBLIOGRAFÍAS

Werbach K. y Hunter D. (2014). Revoluciona tu negocio con las técnicas de los juegos. Madrid, España: Pearson

Oakley B. (2014). Abre tu mente a los números. Barcelona, España: RBA

Gorris, J. M. (1977). El juguete y el juego. Madrid, España: Avance

Deci, E y Ryan R. (2004). Handbook of Self-Determination Research.Reino Unido, Europa: University Rochester Press

Siegal, J.(14, Mayo, 2014). Tiempo medio gastado en videojuegos en EEUU. Recuperado de http://bgr.com/2014/05/14/time-spent-playing-video-games/

Deterding S, Dixon D, Khaled R y Nacke L. (2011). From game design elements to gamefulness: defining gamification. Recuperado de http://gamification-research.org/wp-content/uploads/2011/04/02-Deterding-Khaled-Nacke-Dixon.pdf

Marton F y Sljo R. (11 de mayo, 2011). Educational Psychology. Recuperado de https://onlinelibrary.wiley.com/doi/abs/10.1111/j.2044-8279.1976.tb02980.x

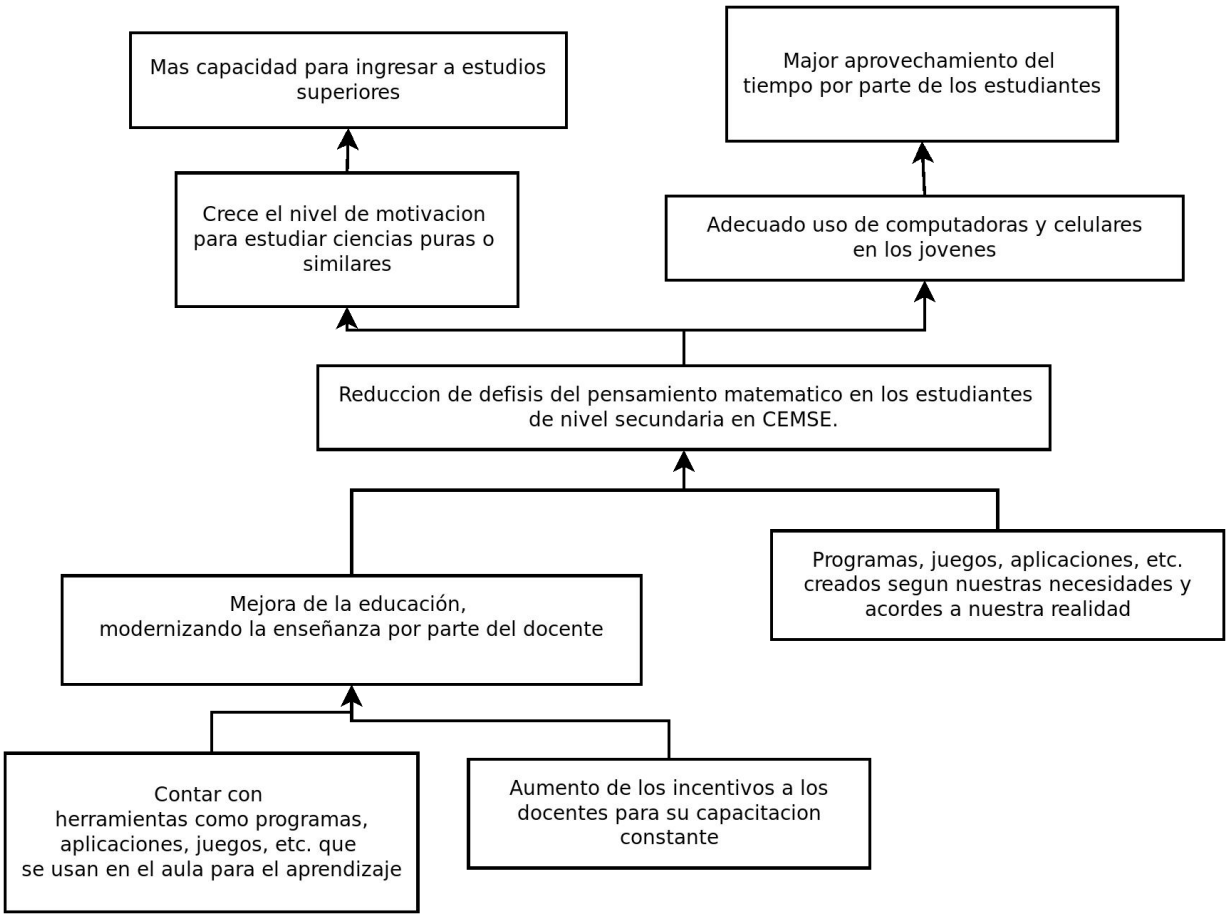
Feldgen, M., y Clua, O. (2004). Games as a motivation for freshman sutdents to leam programming. Fie, 1(1), 1079-1084.

Acerenza, N., Coppers, A., Mesa, G., Viera, A., Fernández, E., Laurenzo, T., & Vallespir, D.(2010). Una Metodología para Desarrollo de Videojuegos. Uruguay.

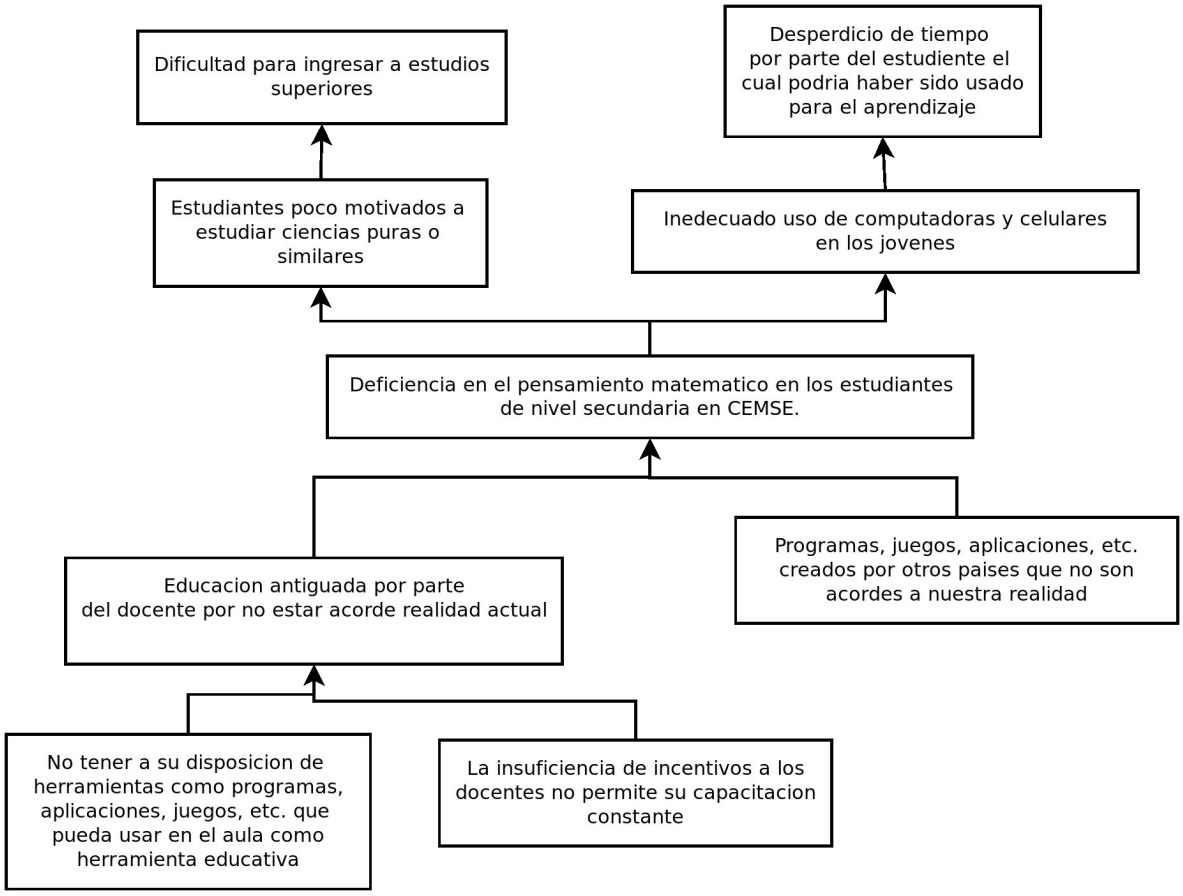
Campayo R. (2006). Desarrolla una mente prodigiosa. España, Europa: Edaf.

ANEXOS

ANEXO A – ÁRBOL DE PROBLEMAS



ANEXO B – ÁRBOL DE OBJETIVOS



ANEXO C – MANUAL DE USUARIO APLICACION

DISEÑO Y DESARROLLO DE UNA APLICACIÓN PARA LA MOTIVACIÓN DEL APRENDIZAJE DE LA MATEMÁTICA UTILIZANDO GAMIFICACIÓN.

CASO: CEMSE

Control de Versiones

Fecha (17/08/2020)

Autor

Felix Benito Mamani Quispe

Descripción

Creación del Manual de USUARIO

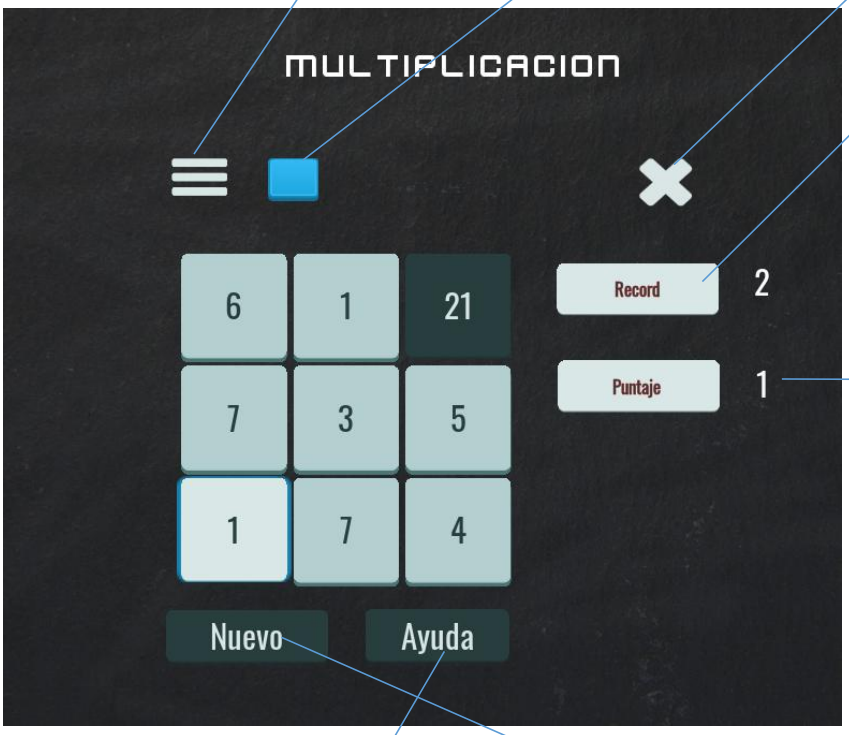
Versión

1.0

Pantalla de inicio y menú principal.

La vista principal de la aplicación, donde se muestra el inicio opciones para ingreso.

Aplicación gamificado



Menú

Acerca del desarrollador

Salir

Cantidad de puntaje acumualdo

Puntaje actual

La casilla 1 deberá moverse por toda la matriz para multiplicar. Hasta llegar al resultado que en este caso es 21. Debera realizar movimiento inteligentes caso contrario perderá.

Ayuda de como realizar cálculos.

Reinicia el juego, para volver a intentarlo.

Herramientas para el calculo mental:

Salir

Agregar nueva casilla de a uno.

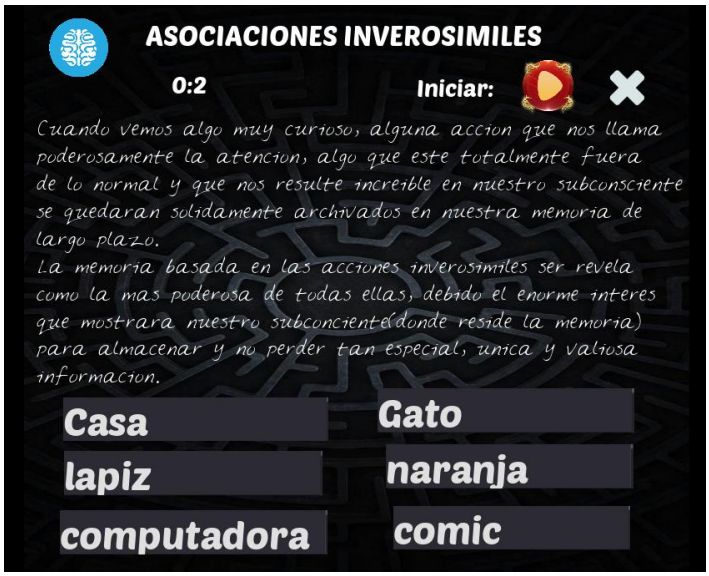
Muestras todas las casillas a guardar. En el cual se habilitara una nuevo botón

Siguiente

Ejercicios asociaciones:

Ejercicio para practicar la Técnica de memoria.

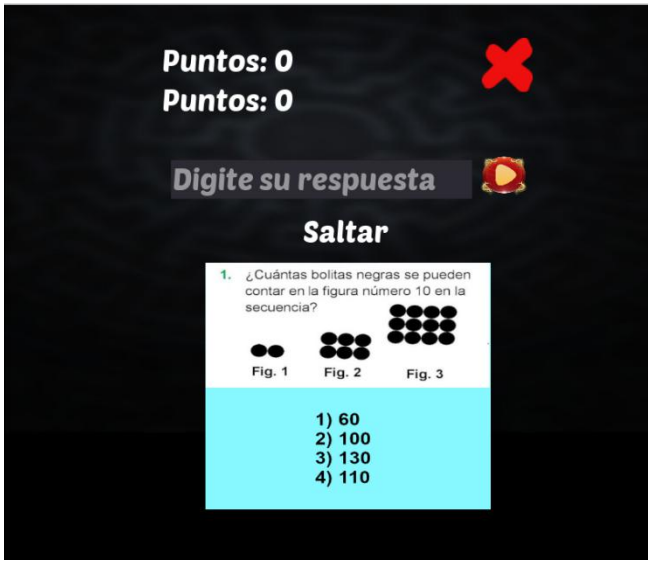
Recordatorios:



Cuando tiene este imagen son conceptos importantes, a modo de recordatorio.

Ejercicios de lateralidad.

Genera automáticamente una serie de letras aleatorias siguiendo ciertas reglas. Para el ejercicio de lateralidad.



Genera automáticamente preguntas que se deberán responder.

Técnica de memoria con números aleatorios

ANEXO D – Código Arduino

```
/*
  REALIDAD VIRTUAL CON ARDUINO
  Código para leer la inclinación del casco RV
*/


#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BNO055.h>
#include <utility/imumaths.h>

//Delay entre lecturas
#define BNO055_SAMPLERATE_DELAY_MS (15)
Adafruit_BNO055 bno = Adafruit_BNO055(55); //Crear el objeto IMU

void setup(void)
{
  Serial.begin(115200); //Activar el serial

  //Activar el sensor
  Serial.println("Activar sensor");
  bno.begin();

  //Delay de tres segundos para dar tiempo a calibrar el magnetometro
  delay(3000);
}

void loop(void)
{
  //Leer la IMU
  sensors_event_t event;
  bno.getEvent(&event);

  /*Los ejes estan girados: el X es el Y, el Y es el Z y el Z es el X
    Ademas, hay que sumar 90 para que el eje X,Y vaya de 0 a 180*/
  float ejeX = (float)event.orientation.y+90;
  float ejeY = (float)event.orientation.z+90;
  float ejeZ = (float)event.orientation.x;

  Serial.print('1'); //Enviar el código de seguridad

  //Como queremos que el angulo se envié con 3 números antes
  //de la coma flotante, hay que ver cuantas cifras tiene.
  //Si tiene 1 cifra (es menor que 10) se escriben dos ceros
  //Si tiene 2 cifras (es menor que 100) se escribe un solo cero
  if(ejeX < 10)
  {
    Serial.print("00");
  }
  else if(ejeX < 100)
```

```
  {
    Serial.print("0");
  }
  Serial.print(ejeX); //Ahora si, escribir el angulo. Por defecto se
escribirá con dos cifras decimales.


  //Lo mismo con el eje Y...
  if(ejeY < 10)
  {
    Serial.print("00");
  }
  else if(ejeY < 100)
  {
    Serial.print("0");
  }
  Serial.print(ejeY);

  //...y el eje Z!
  if(ejeZ < 10)
  {
    Serial.print("00");
  }
  else if(ejeZ < 100)
  {
    Serial.print("0");
  }
  Serial.print(ejeZ);

  //Enviar el final de linea
  Serial.print("\n");

  delay(BNO055_SAMPLERATE_DELAY_MS);
}
```

ANEXO E – Código Python

```
"""
REALIDAD VIRTUAL CON ARDUINO Y GODOT

"""

from godot import exposed, export
from godot.bindings import *
from godot.globals import *

import serial
import math


@exposed
class Main(Node):

    def _process(self, delta):

        #Abrimos el serial a 115200 baudios
        ser = serial.Serial('/dev/ttyUSB1', 115200)
        #Nota: cambia la dirección /dev/ttyUSB0 por la de tu placa
Arduino

        #Leemos una linea y la decodificamos
        a = ser.readline()
        a = a.decode("ascii")


        #Guardamos las dos cabezas
        headL = self.get_node("Head/HBoxContainer/VCL/Viewport/HeadL")
        headR = self.get_node("Head/HBoxContainer/VCR/Viewport/HeadR")

        #Si el mensaje empieza por 1 y tiene un tamaño de 20, es
válido
        if(a[0] == '1' and len(a) == 20):

            #Limpiar la lectura para encontrar el angulo
            a = a[:-1] #Eliminar el ultimo carácter
            a = a[1:] #Eliminar también el primer carácter (1)

            #Guardar los ángulos
            angleX = a[0:6]
            angleY = a[6:12]
            angleZ = a[12:]

            #Convertirlos a float y aplicar correcciones (puede que
tengas que adaptarlas)
            angleX = float(angleX)-90
            angleY = +float(angleY)-90
            angleZ = -float(angleZ)
```

```
            #Calculamos los ángulos de rotación del objeto cabeza
(rotación casco + rotación nave)
            angleX = math.radians(angleX)
            angleY = math.radians(angleY)
            angleZ = math.radians(angleZ)

            #Cambiamos la rotación de las cabezas
            headL.set_rotation(Vector3(angleX, angleZ, angleY))
            headR.set_rotation(Vector3(angleX, angleZ, angleY))

        #Cerrar el serial
        ser.close()
```