UNIVERSIDAD PERUANA DE CIENCIAS APLICADAS

FACULTAD DE INGENIERÍA

PROGRAMA ACADÉMICO DE INGENIERÍA DE SOFTWARE

Game engine y framework open-source para el desarrollo de videojuegos 2D

TESIS

Para optar por el título profesional de Ingeniero de Software

AUTOR(ES)

Miramira Morales, Bryan Antony (0009-0003-6831-8007)

Espinal Campos, Sergio Alejandro (0009-0008-1913-0063)

ASESOR

Velásquez Nuñez, Ángel Augusto (0000-0001-9668-6972)

Lima, 07 de mayo de 2023

DEDICATORIA

A Dios, nuestros padres y hermanos que nos apoyaron en la etapa universitaria y laboral, a ellos por su constante seguimiento y apoyo para la obtención del título universitario.

AGRADECIMIENTOS

Sergio Espinal:

A mi Señor, Dios y Padre Jesucristo por ayudarme en esta etapa universitaria, a mi padre y hermano que me apoyaron, y ayudaron siempre. Asimismo, a todos mis docentes que participaron en mi aprendizaje a lo largo de toda mi carrera profesional.

Bryan Miramira:

A nuestros docentes de la etapa universitaria, asesores de titulación, y especialmente a nuestras familias por su constante apoyo.

RESUMEN

Actualmente, en el campo de desarrollo de videojuegos es común el uso de herramientas sofisticadas que ayudan a acelerar la construcción de un juego, ya sea en 2D o 3D. Usualmente son llamadas Game Engine, el cual es una aplicación de escritorio que provee de una interfaz para la construcción de personajes, escenarios y todas las funcionalidades necesarias para la elaboración de un videojuego totalmente funcional y eficiente. Dicha herramienta permite a los desarrolladores de juegos centrar todos sus esfuerzos en diseñar su idea de juego (trama, personajes, escenarios, jugabilidad, etc.) sin preocuparse en cómo se implementará la lógica a nivel de desarrollo de software (patrones, arquitectura, pipeline, audio, render y otros). Sin embargo, la mayoría de estas herramientas están especializadas en realizar tareas para entornos en 3D, asimismo normalmente son muy genéricos, ya que no cuentan con funcionalidades destinados para géneros de juegos en específicos. Esto ocasiona que los desarrolladores de videojuegos 2D tengan que invertir mucho tiempo en realizar configuraciones en los Game Engines para generar gráficos 2D e implementar funcionalidades para entornos 2D y géneros de juegos en específico, lo cual ocasiona una reducción notable en la productividad, en la publicación de juegos y el riesgo de construir videojuegos con problemas de rendimiento y bugs. Por ello, el presente proyecto propone la elaboración de un Game Engine para la construcción de juegos en 2D que utiliza su propio Game Framework para lograr una mayor flexibilidad en la extensiones de funcionalidades futuras. Asimismo, aplicaremos patrones de diseño y arquitectura especializadas en videojuegos, con el fin de asegurar un alto rendimiento y eficiencia del producto final.[1]

Palabras clave**:** game engine; game framework; game development; 2d games; open-source game development

[1] Este trabajo originalmente fue publicado en Espinal, S., Miramira, B., & Velásquez, Á. (2022). Open-Source Game Engine & Framework for 2D Game Development. *2022 IEEE Engineering International Research Conference (EIRCON)*, 1-4. https://doi.org/10.1109/EIRCON56026.2022.9934816

ABSTRACT

Currently, in the field of video game development it is common to use sophisticated tools that help speed up the construction of a game, either in 2D or 3D. They are usually called Game Engine, which is a desktop application that provides an interface for the construction of characters, scenarios and all the necessary functionalities for the elaboration of a fully functional and efficient videogame. Such a tool allows game developers to focus all their efforts on designing their game idea (plot, characters, scenarios, gameplay, etc.) without worrying about how the logic will be implemented at the software development level (patterns, architecture, pipeline, audio, rendering and others). However, most of these tools are specialized in performing tasks for 3D environments, and they are usually very generic, since they do not have functionalities intended for specific game genres. This causes 2D video game developers to spend a lot of time configuring Game Engines to generate 2D graphics and implement functionalities for 2D environments and specific game genres, which causes a significant reduction in productivity, in the publication of games and the risk of building video games with performance problems and bugs. Therefore, this project proposes the development of a Game Engine for the construction of 2D games that uses its own Game Framework to achieve greater flexibility in the extensions of future functionalities. Likewise, we will apply design and architecture patterns specialized in videogames, in order to ensure high performance and efficiency of the final product.

Keywords: game engine; game framework; game development; 2d games; open-source game development

N°6542_Game engine y framework open-source para el desarrollo de videojuegos 2D

INFORME DE ORIGINALIDAD

11%	11%	1%	6%
INDICE DE SIMILITUD	FUENTES DE INTERNET	PUBLICACIONES	TRABAJOS DEL ESTUDIANTE

FUENTES PRIMARIAS

1	repositorioacademico.upc.edu.pe Fuente de Internet	4%
2	upc.aws.openrepository.com Fuente de Internet	4%
3	Submitted to Universidad Peruana de Ciencias Aplicadas Trabajo del estudiante	2%
4	g-mnews.com Fuente de Internet	<1%
5	Zoë O'Shea, Jonathan Freeman. "Game design frameworks", Proceedings of the 14th International Conference on the Foundations of Digital Games - FDG '19, 2019 Publicación	<1%
6	"HCI in Games: Experience Design and Game Mechanics", Springer Science and Business Media LLC, 2021 Publicación	<1%

7	Submitted to IUBH - Internationale Hochschule Bad Honnef-Bonn Trabajo del estudiante	<1 %
8	hdl.handle.net Fuente de Internet	<1 %
9	1library.co Fuente de Internet	<1 %
10	www.dykinson.com Fuente de Internet	<1 %
11	idoc.pub Fuente de Internet	<1 %

Excluir citas	Apagado	Excluir coincidencias	< 20 words
Excluir bibliografía	Activo		

TABLA DE CONTENIDOS

1. CAPÍTULO 1: DEFINICIÓN DEL PROYECTO 14
1.1 Contexto del problema 14
1.2 Dominio del problema 18
1.3 Planteamiento de la solución 18
1.4 Objetivo del proyecto 19
1.4.1 Objetivo general 19
1.4.2 Objetivos específicos 19
1.5 Indicadores de éxito 19
1.6 Planificación del proyecto 20
1.6.1 Gestión del tiempo 20
1.6.2 Alcance 21
1.6.3 Riesgos y mitigaciones 22
2. CAPÍTULO 2: LOGROS DE STUDENT OUTCOMES 23
2.1 Competencias específicas 23
2.1.1 Student Outcome 1 23
2.1.2 Student Outcome 2 23
2.1.3 Student Outcome 3 24
2.1.4 Student Outcome 4 24
2.1.5 Student Outcome 5 27
2.1.6 Student Outcome 6 28
2.1.7 Student Outcome 7 28
3. CAPÍTULO 3: MARCO TEÓRICO 30
3.1 Game Engine 30
3.2 Game Framework 30
3.3 Middleware 31
3.4 Entity Component System 31
3.5 Scene Management 32
3.6 Arquitectura modular 32
3.7 Dear ImGUI 33
3.8 Scripting 33
3.9 OpenGL 33
3.11 GLAD 34
3.12 GLM 34
3.13 Box2D 34

4. CAPÍTULO 4: DESARROLLO DEL PROYECTO 35
4.1 Elaboración de Benchmarking 35
4.1.1 Benchmarking sobre las características de los Game Engines 35
4.1.2 Benchmarking sobre las características de los Game Frameworks 36
4.1.2 Benchmarking sobre los lenguajes de programación para el desarrollo de videojuegos 37
4.1.3 Análisis de Benchmarking 37
4.2 Arquitectura del proyecto 38
4.2.1 Arquitectura Física 38
4.2.2 Arquitectura Lógica 38
4.2.3 Vista de Contexto del Sistema 39
4.2.4 Vista de Contenedor 40
4.2.5 Vista de Componente 41
4.2.6 Vista de Código 45
4.3 Game Engine y Game Framework 46
5. CAPÍTULO 5: RESULTADOS DEL PROYECTO 50
5.1 Pruebas de aceptación 50
5.1.1 Validación con novatos 50
5.1.2 Validación con expertos 50
5.2 Aplicación de encuesta 51
5.2.1 Cuestionario 51
5.3 Análisis de resultados 61
6. CAPÍTULO 6: GESTIÓN DEL PROYECTO 64
6.1 Plan de gestión del Proyecto 64
6.1.1 Fases del proyecto 64
6.1.2 Enfoques de desarrollo 66
6.1.3 Planes de Gestión Subsidiaria 66
6.1.4 Umbral de Variación de Alcance 67
6.1.5 Gestión del Alcance de la Línea Base 67
6.1.6 Umbral de Variación del Calendario 67
6.1.7 Gestión de Línea de Base del calendario 68
6.1.8 Umbral de Variación de Costos 68
6.1.9 Gestión de la línea Base de los Costos 68
6.2 Plan de gestión de alcance 69
6.2.1 EDT 69
6.2.2 Diccionario EDT 70
6.2.3 Alcance de la línea base de Mantenimiento 71

6.2.4 Aceptación de Entregable 71
6.2.5 Alcance e Integración de requisitos 71
6.3 Plan de gestión de requisitos 71
6.3.1 Recolección de Requisitos 71
6.3.2 Análisis de Requisitos 71
6.3.3 Categorías de Requisitos 72
6.3.4 Documentación de Requisitos 72
6.3.5 Priorización de Requisitos 72
6.3.6 Métricas 72
6.3.7 Estructura de Trazabilidad 73
6.3.8 Seguimiento de Requisitos 73
6.3.9 Requisitos de Reporting 73
6.3.10 Validación de Requisitos 73
6.3.11 Gestión de la Configuración de Requisitos 73
6.4 Plan de gestión de cronograma 74
6.4.1 Metodología de Asignación del Cronograma 74
6.4.2 Herramientas de Planificación y Asignación del Cronograma 74
6.4.3 Nivel de Exactitud 74
6.4.4 Unidades de Medida 74
6.4.5 Umbrales de Variación 74
6.4.6 Asignación de Informes y Formatos 74
6.4.7 Enlaces de Procedimientos Organizacionales 75
6.4.8 Actualización del Cronograma 75
6.5 Lista de Hitos 75
6.6 Plan de gestión de comunicaciones 76
6.6.1 Matriz de comunicaciones 76
6.6.2 Restricciones o Suposiciones de Comunicación 77
6.6.3 Glosario de Terminología Común 77
7. CONCLUSIONES 78
8. RECOMENDACIONES 80
9. [ANEXOS] 81
Anexo A: WASC 81
Anexo B: Atributos de calidad 87
Anexo C: Costos y presupuestos 90
10. REFERENCIAS BIBLIOGRÁFICAS 94

ÍNDICE DE TABLAS

Tabla 1 Problema identificado y sus causas 17
Tabla 2 Hitos del proyecto 21
Tabla 3 Riesgos del proyecto 22
Tabla 4 Benchmarking sobre características de los Game Engines 35
Tabla 5 Benchmarking sobre características de los Game Frameworks 36
Tabla 6 Benchmarking sobre los lenguajes de programación para desarrollar videojuegos 37
Tabla 7 Niveles de escala según el tipo de respuesta 51
Tabla 8 Fases del proyecto 64
Tabla 9 Enfoques de desarrollo 66
Tabla 10 Planes de gestión subsidiaria 66
Tabla 11 Lista de hitos 75
Tabla 12 Matriz de comunicaciones 76
Tabla 13 Restricciones o suposiciones de comunicación 77
Tabla 14 Tácticas 88
Tabla 15 Tácticas por atributo de calidad 89
Tabla 16 Costos de infraestructura 91
Tabla 17 Gastos 91
Tabla 18 Métodos de ingreso 92
Tabla 18 Indicadores de coste 92

ÍNDICE DE FIGURAS

Figura 1 Tendencias en miles de millones de dólares entre música, películas y videojuegos 14
Figura 2 Número de juegos independientes lanzados en la plataforma Steam a nivel mundial. 15
Figura 3 Ventas de juegos indie por precio en el 2020. 16
Figura 4 Juegos indie lanzados en Steam por género y ganancias en el 2020 16
Figura 5 Perspectiva de la arquitectura física para el desarrollador y el jugador 38
Figura 6 Vista general del proyecto 39
Figura 7 Vista de contenedores del proyecto 40
Figura 8 Vista de componentes del proyecto (Completo) 41
Figura 9 Vista de componentes del proyecto (DEV GUI) 42
Figura 10 Vista de componentes del proyecto (Game Engine) 42
Figura 11 Vista de componentes del proyecto (Game Framework) 43
Figura 12 Vista de componentes del proyecto (2D Video Game) 43
Figura 13 Vista de componentes del proyecto (Game Generator) 44
Figura 14 Vista de código del proyecto 45
Figura 15 Interfaz del Game Engine 46
Figura 16 Inspector de propiedades del Game Engine & Framework 47
Figura 17 Content browser de los Game Objects 47
Figura 18 Content browser del Sprite sheet 48
Figura 19 Gizmo en los Game Objects 48
Figura 20 Animation Editor para el player 49
Figura 21 Player y Game Objects 49
Figura 22 Resultados de interacción 52
Figura 23 Resultados del diseño 53
Figura 24 Resultados de la creación de Game Objects 53
Figura 24 Resultados de la creación de Sprite sheets 54
Figura 26 Resultados de la física del juego 54
Figura 27 Resultados del inspector de componentes 55
Figura 28 Resultados de Gizmo 55
Figura 29 Resultados del Animation Editor 56

Figura 30 Resultados de las animaciones por evento 57
Figura 31 Resultados del editor de componentes 57
Figura 32 Resultados de la manipulación de Game Objects 58
Figura 33 Resultados de la cámara del juego 58
Figura 34 Resultados de Game & Editor Scene 59
Figura 35 Resultados de los componentes preconfigurados 59
Figura 36 Resultados del archivo de guardado de datos 60
Figura 37 Resultados sobre añadir el proyecto al set de herramientas 60
Figura 37 EDT 69
Figura 39 Taxonomía 87
Figura 40 Ganancia 92
Figura 41 Retorno de inversión 93

1. CAPÍTULO 1: DEFINICIÓN DEL PROYECTO

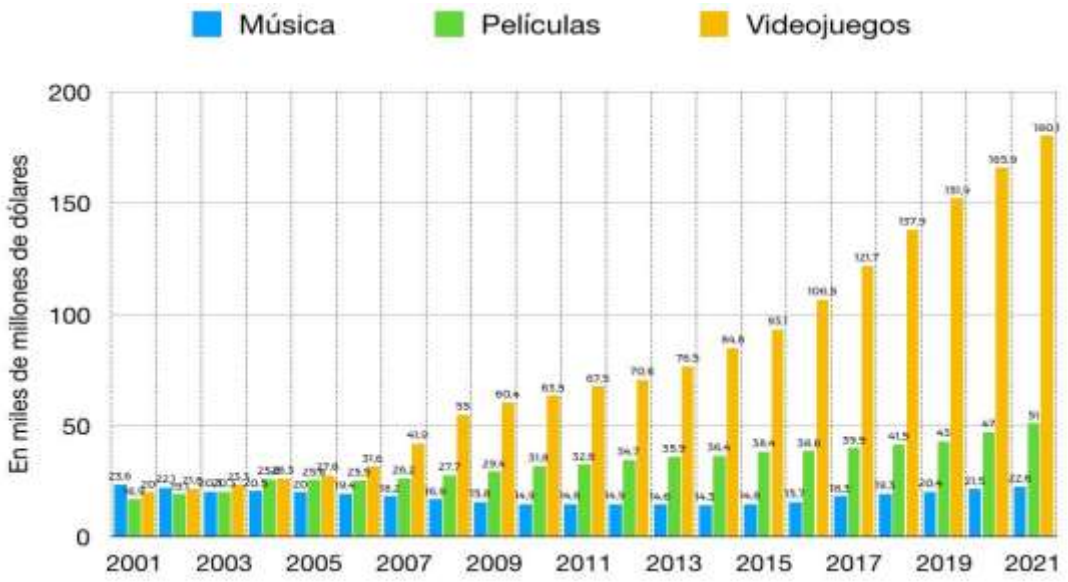
En el presente documento se describirá el contexto y dominio del problema de la siguiente tesis, también, el objetivo general y los específicos. Además, se abordará la planificación del proyecto con sus respectivas secciones como, hitos, alcance, restricciones y mitigaciones.

1.1 Contexto del problema

Actualmente, los videojuegos poseen una increíble popularidad, al nivel que se estima que para el 2022 esta industria obtendrá ingresos por más de 200 mil millones de dólares solo en videojuegos 2D (Northglen News, 2020). Asimismo, la disponibilidad de internet favorece en la distribución y publicación de juegos a nivel mundial, generando así un incremento de ingresos millonario en ganancias. Esto ha permitido que compañías como, Google, Microsoft, Sony, entre otras participen en dicha industria, tanto en el desarrollo como en la distribución de videojuegos 2D y 3D (Interxion, 2019). Además, otros motivos del incremento de la popularidad de los videojuegos son el trabajo en casa, los torneos internacionales con grandes premios y la monetización del streaming de videojuegos (Bryant & Saiedian, 2021).

Figura 1

Tendencias en miles de millones de dólares entre música, películas y videojuegos



Nota. De “Si hay una industria que no es un juego, esa es la de los Videojuegos”, por Cortizo, 2018 (https://en.digital/blog/videojuegos-industria-mobile-crecimiento)

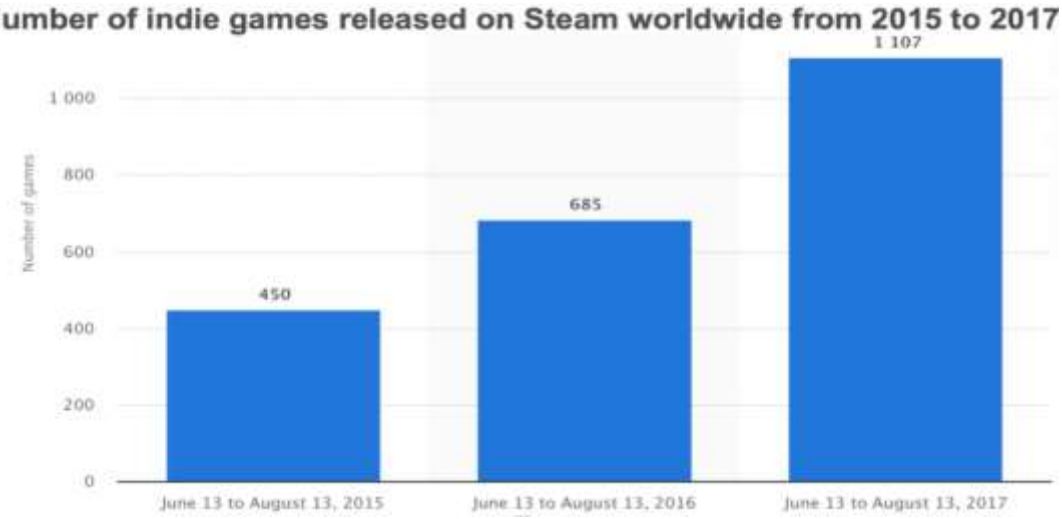
Según Castrejón y Aliaga (2020) en Latinoamérica, los videojuegos tendrán un crecimiento anual del 10% hasta llegar a US$3.580 millones para el 2023. Esto en su mayoría se debe al alto consumo de juegos por parte de Brasil y México. En el caso de Brasil consume US$1.566 millones y México US$1.172 millones. Asimismo, para los videojuegos de PC, se proyecta un ascenso del 7,71% y en Perú 4,91% (G&M News, 2020).

G&M News (2020) indica que el Perú presenta una tendencia de incremento de 20% por año hasta generar US$147 millones para 2023. Además, la industria recolectó US$58 millones en el 2018, lo cual genera grandes expectativas para el mercado peruano. En cantidad de gamers, Perú está en el puesto 49 a nivel mundial con unidad de máquinas compradas de 8 millones, entre ellas consolas, portátiles y computadoras. En el sector de desarrollo de videojuegos Perú generó S/7,1 millones con 18 compañías, de las cuales 13 tienen este trabajo como principal, y las otras 5 como trabajo secundario.

En el sector independiente, el desarrollo de videojuegos ha incrementado en los últimos años. La producción de indie games desde el 2015 al 2017 creció en 40.65% (Statista, 2017). Sin embargo, dicho sector está formado en su mayoría por equipos pequeños de desarrollo, lo cual hace complicado construir juegos 2D con game engines y frameworks del mercado actual, ya que estos están orientadas a videojuegos 3D, y contienen muchas opciones que generan una curva de aprendizaje compleja (Pavkov et al., 2017).

Figura 2

Número de juegos independientes lanzados en la plataforma Steam a nivel mundial



Nota. De “Number of indie games released on Steam worldwide from 2015 to 2017”, por Statista, 2017 (https://www.statista.com/statistics/809258/number-indie-games-steam/)

Figura 3

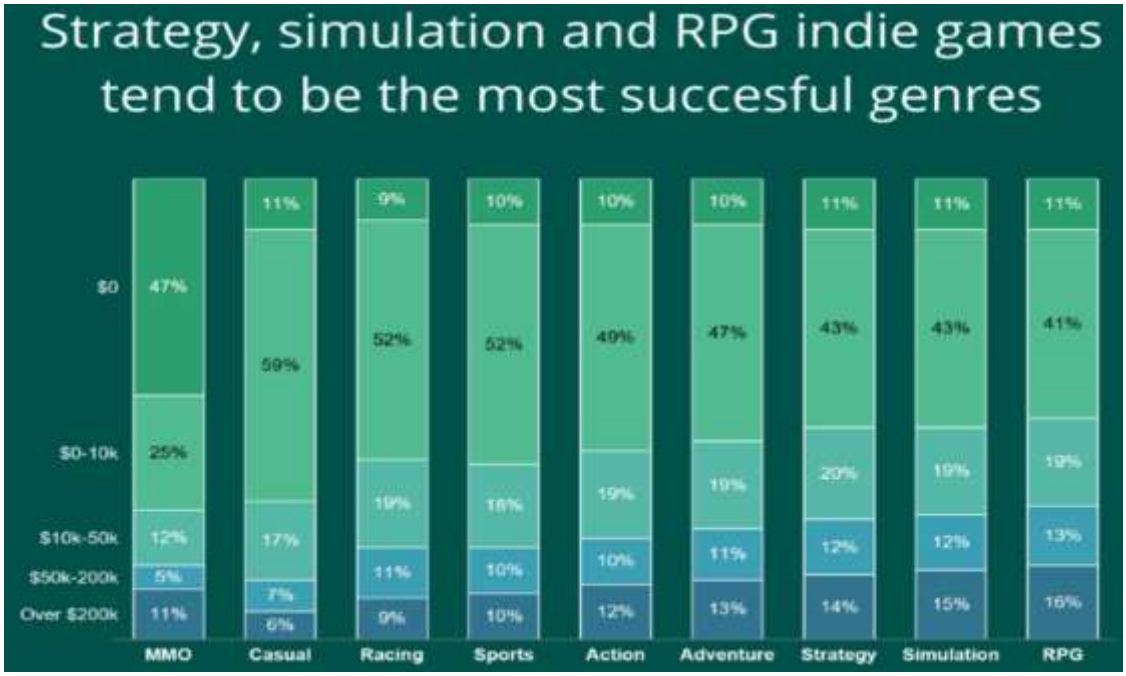
Ventas de juegos indie por precio en el 2020

Price	# of Indie Games	Median Sales
$0.01-$4.99	11,911	$ 1,991
$5-9.99	5,653	$ 11,026
$10-19.99	3,561	$ 38,674
$20-29.99	501	$ 268,111
$30-49.99	187	$ 362,927
Over $50	175	$ 275,972

Nota. De “Are indie games priced too low?”, por VG Insights, 2020a (https://vginsights.com/insights/article/should-developers-charge-more-for-an-indie-game)

Figura 4

Juegos indie lanzados en Steam por género y ganancias en el 2020



Nota. De “Infographic: Indie game revenues on Steam”, por VG Insights, 2020b (https://vginsights.com/insights/article/infographic-indie-game-revenues-on-steam)

Tabla 1

Problema identificado y sus causas

Problema	**Causas**
La oferta de game engines para videojuegos 2D open source es limitada, por lo cual los desarrolladores tienden a utilizar e integrar por cuenta propia diversas configuraciones, herramientas y bibliotecas de distintas fuentes para construir este tipo de videojuegos.	∉ Los game engines del mercado actual en su mayoría son genéricos (Pavkov et al., 2017), por ello el desarrollador tiene que integrarse con una serie de características que no va a utilizar. ∉ Los game engines usualmente poseen una interfaz y funcionalidades enfocado en ambientes con gráficos en 3D, lo cual incrementa la complejidad al momento de desarrollar videojuegos con escenarios 2D. ∉ Los game engines de propósito general tienen optimizado técnicas de IA del tipo collision detection y pathfinding para entornos 3D (Petridis et al., 2012), lo cual hace que no sean útiles en escenarios 2D y que se tengan que implementar una versión 2D por los mismos desarrolladores. ∉ Los game engines genéricos al enfocarse en ambientes 3D, necesitan cargar capas adicionales, lo que causa un consumo de recursos innecesarios para desarrolladores de juegos 2D.

1.2 Dominio del problema

El presente proyecto aborda la necesidad de contar con un Game Engine y Framework que permita el desarrollo de videojuegos en dos dimensiones (2D) de forma nativa y eficiente. Asimismo, que ayude a reducir en tiempo y en esfuerzo la construcción de juegos en 2D para permitir a los desarrolladores enfocarse en la idea de juego y no el detalle de la implementación de las funcionalidades de la aplicación. Esta necesidad surge a partir que en el mercado actual es complicado hallar Game Engines centrados en entornos 2D, ya que la mayoría está orientado a ambientes en 3D. Esto implica que los desarrolladores 2D tengan que invertir un tiempo considerable en realizar configuraciones de conversión de gráficos 3D a 2D o utilicen bibliotecas para construir videojuegos desde cero. Dicho esfuerzo extra disminuye la productividad e introduce problemas de rendimiento y bugs. Esto provoca que los autores publiquen pocos juegos y que muchos de estos sean de baja calidad debido a temas de rendimiento, optimización, poca elaboración del argumento y mala jugabilidad.

1.3 Planteamiento de la solución

Se ha planeado el desarrollo de un Game Engine para videojuegos 2D que utilizará una serie de interfaces abstractas definidas en un Game Framework. Esto permite una mejor personalización y extensión de las funcionalidades del proyecto.

El Game Framework tendrá todas las características necesarias para desarrollar un juego 2D como, módulos de gráficos, sonido, administrador de archivos, física, input, sistemas, componentes y entidades. Esto permitirá aumentar las funcionalidades y/o agregar mejoras o arreglos a las existentes sin necesidad de modificar la arquitectura o dañar el código fuente estable del Game Engine.

El Game Engine implementará y/o extenderá los módulos del Game Framework, de esta manera el desarrollador podrá construir un videojuego 2D a través de una interfaz gráfica. Esto le permitirá realizar configuraciones o definiciones de atributos del escenario, objetos y personajes del juego sin realizar programación, lo cual ayudará a mantener un alto rendimiento, menos errores y más productividad en el proceso de construcción del videojuego 2D.

Finalmente, al ser un proyecto open-source, la comunidad tendrá la libertad de aportar nuevos módulos para ampliar las características del Game Engine y del Game Framework. Además, el código fuente estará completamente escritos en inglés para una mejor globalización. Asimismo, el proyecto será multiplataforma en sistemas operativos como, Windows, Linux y macOS.

1.4 Objetivo del proyecto

1.4.1 Objetivo general

Implementar un Game Engine y Framework que permita el desarrollo de videojuegos 2D.

1.4.2 Objetivos específicos

Para poder cumplir con el objetivo general descrito anteriormente, proponemos los siguientes objetivos específicos:

- **OE1:** Analizar las características de los Game Engines y Game Frameworks, así como los lenguajes de programación para su desarrollo.
- **OE2:** Diseñar un Game Engine y Game Framework que tengan las características típicas que permitan el desarrollo de un videojuego 2D y la arquitectura tanto física como lógica para este proyecto.
- **OE3:** Implementar un Game Engine y Game Framework que permita el desarrollo de videojuegos 2D.
- **OE4:** Desarrollar un videojuego 2D mediante el uso del Game Engine y Game Framework implementado.

1.5 Indicadores de éxito

El cumplimiento de los objetivos del proyecto se mide a través de los siguientes indicadores de logro:

Indicadores de éxito del Objetivo 1:

- **OE1-I1.** Acta de conformidad del Product Owner (PO) del documento de benchmarking respecto a los lenguajes de programación para el desarrollo de un Game Engine y Game Framework.

- **OE1-I2.** Acta de conformidad del Product Owner (PO) del documento de benchmarking respecto a las características de los Game Engines y Game Frameworks que están disponibles en el mercado actual.

Indicadores de éxito del Objetivo 2:

- **OE2-I1.** Acta de conformidad con la aprobación del Product Owner (PO) sobre el lenguaje de programación y bibliotecas a usar en el proyecto. Asimismo, las características mínimas definidas para el Game Engine y Game Framework.

- **OE2-I2.** Acta de conformidad con la aprobación del Product Owner (PO) del diseño de la arquitectura lógica y física del proyecto propuesto.

Indicadores de éxito del Objetivo 3:

- **OE3-I1**. Acta de conformidad con la aprobación del Product Owner (PO) de la implementación del Game Engine 2D y Game Framework 2D que permita el desarrollo de videojuegos 2D.

Indicadores de éxito del Objetivo 4:

- **OE4-I1.** Acta de conformidad con la aprobación del Product Owner (PO) sobre el videojuego 2D desarrollado, el cual cumple con los requisitos establecidos.

- **OE4-I2.** Acta de conformidad con la aprobación del Product Owner (PO) sobre las validaciones realizadas con los desarrolladores expertos y novatos para definir el éxito del proyecto.

1.6 Planificación del proyecto

1.6.1 Gestión del tiempo

Tabla 2

Hitos del proyecto

Hito	Fecha Estimada	Descripción del Hito	Tipo
Hito 1: Aprobación de Project Charter	13/04/2021	Validación del Project Charter con el Product Owner y el Portfolio Manager.	Obligatorio
Hito 2: Objetivo 1 versión final	04/05/2021	Validación del Objetivo 1 con el Product Owner y el Portfolio Manager.	Obligatorio
Hito 3: Objetivo 2 versión final	23/06/2021	Validación del Objetivo 2 con el Product Owner y el Portfolio Manager.	Obligatorio
Hito 4: Objetivo 3 versión final	21/10/2021	Validación del Objetivo 3 con el Product Owner y el Portfolio Manager.	Obligatorio
Hito 5: Cartera de proyectos versión final	22/10/2021	Validación de la cartera de proyectos con el Product Owner y el Portfolio Manager.	Obligatorio
Hito 6: Objetivo 4 versión final	22/11/2021	Validación del Objetivo 4 con el Product Owner y el Portfolio Manager.	Obligatorio

1.6.2 Alcance

El alcance del proyecto incluirá:

- Documentación de la investigación de las tecnologías que se utilizarán para el desarrollo del software final.
- Documentación del desarrollo del software, en el cual se incluirá los detalles de la construcción del videojuego 2D y un manual de usuario.
- Implementación de un Game Engine y Game Framework open-source que permite desarrollar videojuegos 2D.
- Desarrollo de un videojuego 2D, el cual servirá para validar el Game Engine y Game Framework open-source construida.
- El videojuego 2D será instalado en un desktop / laptop.

Las exclusiones del proyecto son las siguientes:

- El Game Engine y Game Framework no incluirá Mobile SDK y Console SDK.
- El Game Engine y Game Framework no incluirá conexiones multijugador en línea.
- No se implementarán los módulos, audio, inteligencia artificial, render, GUI e input desde cero, se utilizará bibliotecas open-source disponibles.
- No se utilizarían tecnologías de desarrollo y modelamiento 3D para la construcción del proyecto.

1.6.3 Riesgos y mitigaciones

Tabla 3

Riesgos del proyecto

Riesgo	Probabilidad	Impacto	Estrategia de mitigación
Cumplir los entregables en las fechas establecidas	Baja	Alto	Realizar un análisis de las actividades y priorizar las más importantes
Información insuficiente para el proyecto	Media	Alto	Ampliar la búsqueda de información en más bases de datos académicas y libros
Permanencia de miembros del equipo	Baja	Alto	Notificar a la PMO para un análisis e implementación de una solución eficaz
Dificultad para encontrar desarrolladores de videojuegos	Media	Alto	Notificar al PO y PM para plantear una estrategia que nos permite contactarnos con desarrolladores de videojuegos
Dificultad para terminar la implementación del proyecto a tiempo	Media	Alto	Evaluar con el PO la posibilidad de reducir el alcance de funcionalidades del proyecto

2. CAPÍTULO 2: LOGROS DE STUDENT OUTCOMES

2.1 Competencias específicas

2.1.1 Student Outcome 1

Demuestra capacidad de identificar, formular y resolver problemas complejos aplicando los principios de ingeniería, ciencia y matemática.

En la fase inicial del proyecto, se identificó la necesidad de contar con un Game Engine y Framework para el desarrollo de videojuegos en 2D, ya que la oferta actual nos brinda en su mayoría herramientas para entornos en 3D, lo cual dificulta el desarrollo y productividad en el lanzamientos de nuevos juegos. Esto se explica a mayor detalle en el capítulo 1, en las secciones “1.1 Contexto del problema” y “1.2 Dominio del problema”.

La solución del problema es planteada a partir de las causas identificadas y de los estándares en desarrollo de herramientas para la construcción de videojuegos en la industria actual de videojuegos. Dicha solución se describe de forma detallada en el capítulo 1 en la sección “1.3 Planteamiento de la solución” y la implementación en el capítulo 4.

Finalmente, se usaron los principios de matemáticas para el cálculo de los tiempos y costes del proyecto. Asimismo, se usó matemática para la implementación de funcionalidades de computación gráfica como, el renderizado de sprites, física y manejo de input. Por ejemplo, definición de matriz identidad, multiplicación de matrices, vectores y fuerzas como, gravedad o velocidad. Esto lo podemos ver en el capítulo 4, sección “4.3 Game Engine y Game Framework”, donde se muestra la interfaz de nuestro proyecto, el cual permite realizar cambios en la geométrica de los objetos del juegos, así como modificar las propiedad de físicas de los cuerpos, entre otros.

2.1.2 Student Outcome 2

Demuestra capacidad de aplicar el diseño de ingeniería de Software para producir soluciones que satisfagan necesidades específicas con consideración de salud pública, seguridad y bienestar, así como factores globales, culturales, sociales, ambientales y económicos.

Para nuestro proyecto se utilizaron patrones de software aplicados a videojuegos para el desarrollo del Game Engine y Game Framework. Además, se seleccionó los estilos de arquitectura y su correcta implementación basado en las buenas prácticas de la industria actual de videojuegos. Por ejemplo, se usó patrones de software como, MVC, observer, singleton, template, entre otras. También, enfoques de arquitectura de software como, middleware, modular, scene management y ECS. Dichas definiciones son ampliadas en el capítulo 3. Además, la arquitectura lógica fue diagramada mediante el modelo C4, el cual se ve a detalle en el capítulo 4, sección "4.2.2 Arquitectura Lógica".

Nuestro proyecto, brinda a los desarrolladores de videojuegos 2D una herramienta robusta, con alto rendimiento y confiable para la construcción de sus juegos. Asimismo, al ser un proyecto open-source ayudará a la comunidad de desarrolladores independientes, ya que podrán construir videojuegos 2D de forma gratuita.

Finalmente, este proyecto puede ser fácilmente mantenido y extendido, ya que su arquitectura soportar dichas características y el idioma completo del proyecto es en inglés, lo cual facilita su uso global.

2.1.3 Student Outcome 3

Demuestra capacidad de comunicarse efectivamente con un rango de audiencias.

La comunicación tanto oral como escrita fue muy importante en el transcurso del proyecto, ya que nos permitió intercambiar ideas como equipo, realizar presentaciones a nuestro PM y PO para correcciones, mejoras y retroalimentación en todas las fases del proyecto. Estas reuniones fueron calendarizadas para llevar un orden en cada reunión, donde presentamos el avance de las tareas y recibimos retroalimentación efectiva, tal como se muestra en el capítulo 6, sección "6.6 Plan de gestión de comunicaciones".

2.1.4 Student Outcome 4

Demuestra capacidad de reconocer responsabilidades éticas y profesionales en ingeniería de Software y emite juicios informados, que deben considerar el impacto de las soluciones de ingeniería en contextos globales, económicos, ambientales y sociales.

Durante el desarrollo de nuestro proyecto hemos puesto mucha atención con el cumplimiento ético y profesional, los cuales se pueden evidenciar según el Código de Ética y Práctica Profesional de la Ingeniería de Software (ACM Ethics, 2016).

- **Principio 1: Sociedad.** Para asegurar la construcción de un Game Engine y Framework que sea realmente útil para los desarrolladores de videojuegos 2D, el cual cumpla con los estándares tecnológicos de la industria de videojuegos como, patrones de diseño, estilos de arquitectura, consumo de recursos y lenguajes de programación, se realizó tres tipo de benchmarking, para lenguajes de programación, funcionalidades principales de los Game Engines y Frameworks del mercado actual, como se observa en el capítulo 4, sección "4.1 Elaboración de Benchmarking" . Asimismo, se realizó una investigación sobre enfoques de arquitectura de software orientado a videojuegos basado en trabajos previos (papers) y en la lectura de libros especializados en el tema, tal como se muestra en el marco teórico descrito en el capítulo 3 y capítulo 4, sección "4.2 Arquitectura del proyecto".
- **Principio 2. Cliente y empresario.** Al momento de definir el alcance del proyecto se excluyeron algunas funcionalidades y tecnologías como, el uso de renderizado 3D, así como la construcción de bibliotecas de física, render, sonido, input, entre otras desde cero, ya que utilizaríamos bibliotecas listas para realizar esas tareas. Esto con el fin de enfocarnos en los estilos de arquitectura, los patrones de diseño y la documentación, los cuales aseguren un proyecto eficiente y robusto. Esto se evidencia, en el capítulo 1, sección "1.6.2 Alcance".
- **Principio 3. Producto.** Para desarrollar un producto de calidad y útil para los desarrolladores de videojuegos, se realizó un proceso complejo de búsqueda de las tecnologías, patrones de diseño y estilos de arquitectura idóneos para el proyecto, esto mediante la técnica de benchmarking (revisar capitulo 4, sección 4.1 Elaboración de Benchmarking), investigaciones de nuevos conceptos, herramientas e implementación como se describe en los capítulos 3 y 4. Finalmente, basado en dichos conocimientos, estimación de complejidad inicial y la experiencia del equipo, se procedió a realizar un calendario con fechas adecuadas y con un alcance con exclusiones apropiadas.

- **Principio 4. Juicio.** La selección de las tecnologías, patrones de diseño y estilos de arquitectura se realizó tomando en cuenta siempre la elaboración de un producto estable, robusto y útil para los desarrolladores. Esto implicó aplicar conocimientos avanzados y complejos de C++, así como aprender a profundidad conceptos y estilos de arquitectura aplicado a videojuegos, lo cual no fue un obstáculo para tomar las decisiones correctas y no ir por lo más sencillo o simple. Asimismo, se descartó bibliotecas o herramientas de uso no comercial, aunque implicará un mayor esfuerzo en implementación de funcionalidades. Esto con el objetivo de no generar un conflicto de interés con bibliotecas no permisivas o de uso no comercial cuando los usuarios utilicen nuestro proyecto para elaborar videojuegos comerciales.
- **Principio 5. Administración.** Para el desarrollo de la presente tesis, se realizó un análisis de las fases del proyecto, sobre los riesgos con sus respectivas acciones de contingencia, calendario de exposiciones y de entrega, como se puede apreciar en el capítulo 6. Asimismo, se optó por utilizar el marco de trabajo SCRUM para asegurar un ciclo de desarrollo ágil, adaptativo a cambios y con entregas del producto incremental. Además, evaluamos los atributos de calidad que debe contar nuestro proyecto según lo recomendado por Nicholas Graham en su clase de desarrollo de videojuegos y arquitectura, así como las estrategias para implementarlos, como se observa en la sección “Anexo B: Atributos de calidad”.
- **Principio 6. Profesión.** Nuestro Game Engine y Framework tiene como propósito ayudar a los desarrolladores de videojuegos 2D a construir sus propios juegos para distribuirlos de forma comercial o no comercial. Por tanto, nos hemos asegurado de incluir dependencias externas como, bibliotecas, assets y binarios con licencia de uso permisivo o de uso libre comercial. Esto con el objetivo de no ocasionar problemas de licenciamiento a los desarrolladores cuando necesiten vender sus juegos. Asimismo, como se puede ver en el capítulo 3, mencionamos las bibliotecas y herramientas a utilizar en diversas tareas dentro del proyecto, lo cual nos permite ser transparentes al momento de separar los aportes realizados en la presente tesis con las dependencias ya existentes para no atribuirnos derechos o logros que no nos corresponden.
- **Principio 7. Colegas.** Durante el desarrollo de nuestro proyecto hemos reconocido el trabajo y aporte de otros autores al momento de citar adecuadamente

todos los trabajos de investigación previa, webs y libros revisados, así como se han referenciado correctamente en la sección 10. Asimismo, las reuniones entre los miembros del equipos y con los asesores nos ayudó a intercambiar ideas e incrementar la calidad de nuestro proyecto, ya que todas las reuniones fueron realizadas con mucho respeto por las opiniones de cada persona y con el objetivo de aprender unos de otros.

- **Principio 8. Personal.** A lo largo del desarrollo de la presente tesis, hemos adquirido nuevos conocimientos en el área de arquitectura y patrones de diseño de software aplicados a videojuegos. Además, como equipo hemos participado activamente en el diseño e implementación de nuestro proyecto, lo cual nos ha llevado a incrementar nuestros conocimientos y experiencia, en diagramar arquitectura, aplicar nuevos conocimientos, así como manejar conceptos avanzados de C++.

En cuanto a la aplicación del cuestionario de evaluación del proyecto se siguió las regulaciones ordenadas en la LPDP (Ley de Protección de Datos Personales), tal como se puede revisar en el capítulo 5.

Finalmente, consideramos que nuestro proyecto tiene un impacto positivo de forma global y económica, ya que ofrecemos un proyecto open-source que puede ser utilizado para construir videojuegos 2D de forma gratuita y ser comercializados por sus autores. Asimismo, al estar alojado en un repositorio en Github puede ser compartido y mejorado por cualquier persona en el mundo.

2.1.5 Student Outcome 5

Demuestra capacidad de funcionar efectivamente en un equipo cuyos miembros juntos proporcionan liderazgo, crean un entorno de colaboración e inclusivo, establecen objetivos, planifican tareas y cumplen objetivos.

Para el desarrollo del presente proyecto fue necesario trabajar en equipo, se utilizaron herramientas que permiten la colaboración e interacción del grupo tales como Google Meets, One Drive, Jira y se dividieron las tareas en base a objetivos para la planificación de actividades a realizar, correcciones a tomar en cuenta y coordinación en general.

Se contó con el apoyo de los asesores que estuvieron guiándonos durante los distintos módulos resolviendo nuestras dudas y dándonos retroalimentación en las reuniones establecidas según el calendario definido en el capítulo 6, sección “6.6.1 Matriz de comunicaciones”. Además, con nuestro PO y coautor definimos objetivos de entregas a ser presentadas en cada reunión para una mayor productividad y mejor feedback.

2.1.6 Student Outcome 6

Demuestra capacidad de desarrollar y llevar a cabo la experimentación adecuada, analizar e interpretar datos, usando juicio de ingeniería de Software para sacar conclusiones.

Para el desarrollo de experimentos se tomó como entrada los criterios de aceptación de las funcionalidades definidas en las historias de usuario. Asimismo, se realizó una entrega incremental de la implementación de funcionalidades del Game Engine y Game Framework según lo establecido en el marco de trabajo SCRUM. Esto permitió elaborar plan de pruebas de forma incremental según lo con lo que se iba construyendo, mediante pruebas unitarias y de integración entre el Game Engine, Game Framework y GUI.

Para realizar la validación de nuestro proyecto, se realizó un reto, el cual consiste en construir un videojuego en 27 minutos o menos usando nuestra aplicación y otras del mercado actual. Dicho reto fue documentado en un documento en Word y en un video de demostración, los cuales fueron compartidos a los encuestados juntamente con nuestro Game Engine. Asimismo, luego de una semana se les envió una enlace a una encuesta hecha en Google Forms con dos secciones, la primera enfocada en la interfaz de usuario y la segunda que aborda las funcionalidades. Dicho procedimiento se puede apreciar a mayor detalle en la validación de nuestro proyecto en el capítulo 5.

Finalmente, se realizó un análisis de los datos recolectados de la encuesta realizado a los desarrolladores de videojuegos novatos y expertos. Este análisis ha sido divido en dos secciones, resultados de la interfaz y de funcionalidades, que pueden ser vistas en el capítulo 5, sección “5.2 Aplicación de encuesta”.

2.1.7 Student Outcome 7

Demuestra capacidad de adquirir y aplicar nuevos conocimientos según sea necesario, utilizando estrategias de aprendizaje apropiadas.

Durante el diseño de la solución del proyecto, se realizó tres análisis de benchmarking, con el objetivo de conocer y seleccionar correctamente las funcionalidades mínimas que debería tener nuestro Game Engine y Game Framework, así como el lenguaje de programación para su implementación, los cuales pueden ser visualizadas en el capítulo 4, sección “4.1 Elaboración de Benchmarking”.

Asimismo, se adquirió un amplio conocimiento en el ámbito de desarrollo de Game Engines, Game Frameworks y arquitectura de dichas herramientas. Dicho conocimiento, se obtuvo de fuentes bibliográficas con alto impacto; los cuales nos ayudaron a desarrollar un producto que cumpla con los estándares de calidad tratados en cada aporte. Además, se aprendió sobre la implementación de arquitectura ECS, la técnica Batch rendering y OpenGL. Todo ello, puede ser evidenciado en los capítulos 3 y 4.

3. CAPÍTULO 3: MARCO TEÓRICO

3.1 Game Engine

Un Game Engine es una herramienta que permiten generalizar las características típicas de los distintos tipos de videojuegos para facilitar drásticamente su construcción (Thorn, 2011). Por tanto, dicha herramienta cuenta con diversas funcionalidades y algoritmos que permiten desarrollar videojuegos con alto rendimiento, optimizado y con poco esfuerzo. Además, permite la reutilización de código fuente y funcionalidades, ya que muchos juegos utilizan las mismas características (Andrade, 2015). Por ejemplo, un videojuego del tipo shooter usa render de imágenes, al igual que un juego de peleas.

A continuación, se presenta las características comunes de los Game Engines:

- Render de imágenes 2D y 3D
- Input como, teclado, mouse o mandos
- Física
- Sonido
- Interfaz de Usuario
- Animación 2D y 3D
- Procesamiento de hilos
- Scripting
- Inteligencia artificial
- Red

3.2 Game Framework

Gregory (2018) menciona que un game framework en el contexto de herramientas para videojuegos es un esquema que brinda una aplicación con estructura base operativa que controla todas sus las funcionalidades. Dicha aplicación base puede ser personalizada mediante programación al sobrescribir el comportamiento de sus interfaces abstractas. Wozniewicz (2019) indica que un detalle importante para destacar es la diferencia entre un framework y una biblioteca. Un framework posee la característica llamada inversión de control, mientras que una biblioteca carece de ello. Dicha característica le permite al framework controlar el flujo completo de las funcionalidades, lo cual le da estabilidad en el funcionamiento. En el caso de una biblioteca, el desarrollador es quien decide en qué

momento invocar a ciertos métodos y él mismo tiene el control del flujo completo. Por lo anterior, un Game Framework está especializado en proveer de distintas interfaces abstractas con funcionalidades básicas para el desarrollo de videojuegos. Asimismo, estas interfaces pueden ser personalizadas por los desarrolladores según sus necesidades. Además, los desarrolladores no se preocupan por el control interno de los componentes, ya que de esto se encarga el framework. Por ejemplo, en el Game Framework Ogre, los desarrolladores pueden sobrescribir la lógica del bucle principal de render sin preocuparse en qué momento se invocará o como se administrará, ya que lo hará el framework por sí mismo (Gregory, 2018).

3.3 Middleware

Galvan (2018) menciona que este enfoque es utilizado para construir el núcleo del Game Engine pequeño y fácil de mantener, el cual permite que se agreguen a demanda nuevas características que no existen de forma predeterminada. Las entidades del juego pueden suscribirse al middleware para acceder al estado del programa y agregar nuevas funcionalidades como, red, física o entrada de juegos. Además, otro beneficio de este patrón es el acceso al estado del programa sin utilizar un espacio de nombres específico, lo que permite aislar el código. Asimismo, ayuda a precargar entidades fuera de la escena del juego para ahorrar tiempo de carga y aumentar el rendimiento.

3.4 Entity Component System

Entity Component System (ECS) es un patrón de arquitectura de software que utiliza la programación orientada a datos (DOP) para estructurar el código y los datos de un programa, con el objetivo de separar los datos y el comportamiento, lo que permite agregar y eliminar componentes en tiempo de ejecución (Wiebusch & Latoschik, 2015). El patrón ECS es ampliamente utilizado en el área del desarrollo de videojuegos, ya que permite diseñar módulos, funcionalidades y estructuras de datos en general sin aumentar la complejidad de programación del Game Engine. Además, permite gestionar el uso de la memoria RAM de forma más eficiente que otros enfoques, ya que utiliza direcciones de memoria contiguas y punteros. También, dicho enfoque arquitectónico permite el uso de los niveles de caché de la CPU (L1, L2, L3) que ayuda a acceder a los datos con mayor velocidad (Lin, 2020). Los elementos que define dicho enfoque son: la entidad que tiene su propio identificador único (ID) como un número entero para agrupar un conjunto de

componentes del programa. Los componentes que contienen los datos del programa como, píxeles, posición de objetos o código hexadecimal y que se asocian a entidades para un procesamiento de sus datos a futuro por los sistemas. Finalmente, los sistemas que se encargan de tomar los datos de cada componente del programa para procesarlos mediante la relación de componentes por entidad (Raffaillac & Huot, 2019).

3.5 Scene Management

Enfoque arquitectónico que nos permite administrar todos los objectos presentados en la pantalla de juego, así como los recursos auxiliares de audio, inteligencia artificial, renderizado en segundo plano, entre otros. Esto con el objetivo de gestionar la CPU y la RAM del sistema, ya que podemos decidir los momentos de ejecución de distintas tareas como, reproducción de audio, renderizado de personajes, fondos, enemigos, ejecución de algoritmos de inteligencia artificial o la activación de simulación de física de acuerdo con lo que se quiera mostrar en un momento dado del juego. Asimismo, podemos conocer el estado actual de todo lo que se está ejecutando en el juego para decidir qué acciones se van a realizar como, seleccionar el algoritmo del recolector de basura, si el renderizado es estático o dinámico, la simulación de física, la reproducción de audio. Finalmente, el uso del enfoque de Scene Management es muy importante en el desarrollo de videojuegos, ya que permite gestionar mejor los recursos del sistema, así como mejorar el rendimiento del juego (Video Game Technologies, 2022).

3.6 Arquitectura modular

Este enfoque de arquitectura organiza todas las clases de una aplicación en módulos que realizan un conjunto de tareas relacionadas entre sí que representan una funcionalidad en particular. Esto permite clasificar y manejar distintas funcionalidades de la aplicación de forma ordenada, clara, fácil de mantener y entender. De esta manera se puede separar responsabilidades de la aplicación. Asimismo, un módulo puede depender de otro para su funcionamiento, ya que puede haber un conjunto de clases que realizan tareas comunes que pueden ser usadas por distintos módulos. Por tal razón, se recomienda no excederse en la implementación de módulos para evitar dependencias que sean difíciles de mantener. Sin embargo, si utilizamos correctamente este estilo arquitectónico obtendremos una aplicación con capas separadas con funcionalidades bien definidas, con gran facilidad de ser mantenido y flexible (Consulting Tutisani, 2018).

3.7 Dear ImGUI

Dear ImGui es una biblioteca para el desarrollo de interfaz gráficas para C++, eficiente, portátil y autónoma, ya que no tiene dependencias externas. Asimismo, está centrado en la configuración a bajo nivel, es decir mediante programación, lo que le permite aumentar su rendimiento, flexibilidad y productividad en cualquier proyecto. También es una de las mejores herramientas para el desarrollo de motores de juegos y aplicaciones multiplataformas, gracias a su eficiencia, optimización y programación a bajo nivel (Ocornut, 2014).

3.8 Scripting

Modelo de programación donde no se realiza una compilación del código fuente, sino que se va interpretando cada línea de código. Esto lo hace un poco más lento en comparación al lenguaje compilado. Sin embargo, esta diferencia es cada vez menor, debido a las capacidades más potentes de cómputo y a las practicas avanzadas de scripting (GeeksForGeeks, 2019). Asimismo, este modelo accede a bajo nivel de programación gracias a comando especiales, de esta manera logra la comunicación con el hardware (SWIG, 2019).

3.9 OpenGL

API de gráficos multiplataforma 2D / 3D, utilizada en una gran variedad de aplicaciones desde videojuegos hasta editores u otros tipos de software multimedia. Dicha API es independiente al sistema de ventanas y del sistema operativo, lo cual le brinda autonomía al momento de ser ejecutado. Además, se ha utilizado para construir software de alto rendimiento y con gran atractivo visual. Por ejemplo, realidad virtual, videojuegos, medicina, mecánica, ingeniería y fabricación (Khronos Group, 2017).

3.10 GLFW

GLFW (Graphics Library Framework) es una biblioteca que permite crear ventanas, canvas, contextos y capturar la entrada de usuario. GLFW es normalmente utilizado con OpenGL para el desarrollo de aplicaciones modernas con gráficos 2D y 3D (GLFW, 2022).

3.11 GLAD

GLAD (OpenGL Extension Loading Library) es una biblioteca que permite cargar extensiones de OpenGL como, geometry shaders, textures compression, entre otros en tiempo de ejecución. Esto permite generar efectos especiales de pre y post renderizado de imágenes en 2D y 3D de la aplicación que se está construyendo. Asimismo, GLAD es sencillo de adjuntar a cualquier proyecto ya que solo son 4 archivos y estos son generados de forma personalizada gracias a su generador, el cual puede descargarse desde el repositorio de GitHub o acceder directamente desde su website (Khronos Group, 2021).

3.12 GLM

GLM (OpenGL Mathematics) es una biblioteca de matemáticas para OpenGL basado en la especificación GLSL (OpenGL Shading Language) que provee de clases y funciones para realizar operaciones matemáticas complejas. Algunas de las operaciones que se pueden realizar son matrices, transformaciones, vectores, números aleatorios, entre otros. Asimismo, GLM es muy utilizado en el área de procesamiento de imágenes, simulaciones de física, raytracing / rasterization, entre otros (GLM, 2020).

3.13 Box2D

Box2D es una biblioteca escrita en C++ que permite la simulación de física para videojuegos en 2D. Esto permite que los desarrolladores creen videojuegos con mayor realismo, ya que los personajes se mueven en un mundo más interactivo y con características de un espacio real. Además, la biblioteca Box2D posee funcionalidades para colisiones rectangulares o circulares, escuchadores, definición de cuerpos, mundos, gravedad, velocidad, entre otros, lo cual facilita la simulación de física en videojuegos 2D (Box2D, 2015).

4. CAPÍTULO 4: DESARROLLO DEL PROYECTO

El presente capítulo, explicará todo el proceso realizado para el desarrollo del Game Engine y Game Framework para entornos 2D. Se detallarán las tecnologías y patrones de software empleadas cumplir con los objetivos planificados. Asimismo, se presentará la arquitectura del proyecto vista desde la parte lógica con el modelo C4 y la física tanto desde la perspectiva del desarrollador como del jugador.

4.1 Elaboración de Benchmarking

A continuación, se detallará los tres tipos de benchmarking realizados para conocer las características que debe de contar un Game Engine y Game Framework, así como el lenguaje de programación idóneo para este tipo de proyectos.

4.1.1 Benchmarking sobre las características de los Game Engines

Tabla 4

Benchmarking sobre características de los Game Engines

Características	Unity	Unreal Engine	Godot 2D	Game Maker	Ogre	Cry Engine	Ebitengine
Gráficos	✓	✓	✓	✓	✓	✓	✓
Sonido	✓	✓	✓	✓	✓	✓	✓
Input	✓	✓	✓	✓	✓	✓	✓
Física	✓	✓	✓	✓	✓	✓	X
Usa Game Framework	X	✓	X	X	X	X	X
Manejo de Plugins	✓	✓	✓	✓	✓	✓	X
IA para entornos 2D	X	X	X	✓	X	X	X
Enfocado en videojuegos 2D	X	X	✓	✓	X	X	✓
Open Source	X	X	✓	X	✓	X	✓

Es gratuito	X	X	✔	X	✔	X	✔

Nota. Adaptado de "A framework for game engine selection for gamification and serious games", por Ali & Usman, 2016 (https://doi.org/10.1109/ftc.2016.7821753) y de "Game engines: a survey", por Andrade, 2015 (https://doi.org/10.4108/eai.5-11-2015.150615).

4.1.2 Benchmarking sobre las características de los Game Frameworks

Tabla 5

Benchmarking sobre características de los Game Frameworks

Características	Phaser	GGEZ	Duality2D	Zapnet
Render	✔	✔	✔	X
Sonido	✔	✔	✔	X
Assets manager	✔	X	✔	✔
Física	✔	X	✔	X
ECS	X	X	X	✔
Open Source	✔	✔	✔	✔
Manejo de archivos	✔	X	✔	X
IA para entornos 2D	X	X	X	X
Enfocado en videojuegos 2D	✔	X	X	X
Animation 2D	X	X	✔	✔
Multiplataforma	✔	✔	X	✔

Nota. Adaptado de "Introduction to Game Design, Prototyping, and Development", por Gibson, 2022 (https://www.informit.com/store/introduction-to-game-design-prototyping-and-development-9780136619963) y de "Game design frameworks: where do we start?", por O'Shea & Freeman, 2019 (https://doi.org/10.1145/3337722.3337753).

4.1.2 Benchmarking sobre los lenguajes de programación para el desarrollo de videojuegos

Tabla 6

Benchmarking sobre los lenguajes de programación para desarrollar videojuegos

Características	C++	Java	Python	JavaScript	C#	Go
Programación orientada a objetos	✔	✔	✔	X	✔	X
Programación a bajo nivel	✔	X	X	X	X	X
Gestión de la memoria	✔	X	X	X	X	X
Lenguaje compilado	✔	X	X	X	✔	✔
Multiplataforma	✔	✔	✔	✔	X	✔
Alto rendimiento en el desarrollo de videojuegos	✔	X	X	X	✔	X
Usado en la construcción de Game Engines	✔	X	X	✔	X	✔

Nota. Adaptado de "Comparison of Programming Languages in Game Development", por Masood & Wijdan, 2020 (https://www.researchgate.net/publication/342804594_Comparison_of_Programming_Languages_in_Game_Development).

4.1.3 Análisis de Benchmarking

Los benchmarking descritos anteriormente muestran que el Game Engine debe contar con módulos para el manejo de imágenes, sonido, input de dispositivos, simulación de física, ECS, inteligencia artificial y animación. En el caso del Game Framework, debe contar con características de gestión de imágenes, sonido, física y assets. Asimismo, debe administrar módulos para ECS, animación en 2D, inteligencia artificial para entornos 2D y ser multiplataforma. Finalmente, el lenguaje de programación seleccionado es C++, ya que tiene un alto rendimiento en el de desarrollo de videojuegos, dado que permite la programación a bajo nivel para realizar tareas de administración de memoria y de los niveles de caché de la CPU. Asimismo, C++ al ser compilado, su ejecución es mucho más rápida que los lenguajes interpretados o pre-compilado.

4.2 Arquitectura del proyecto

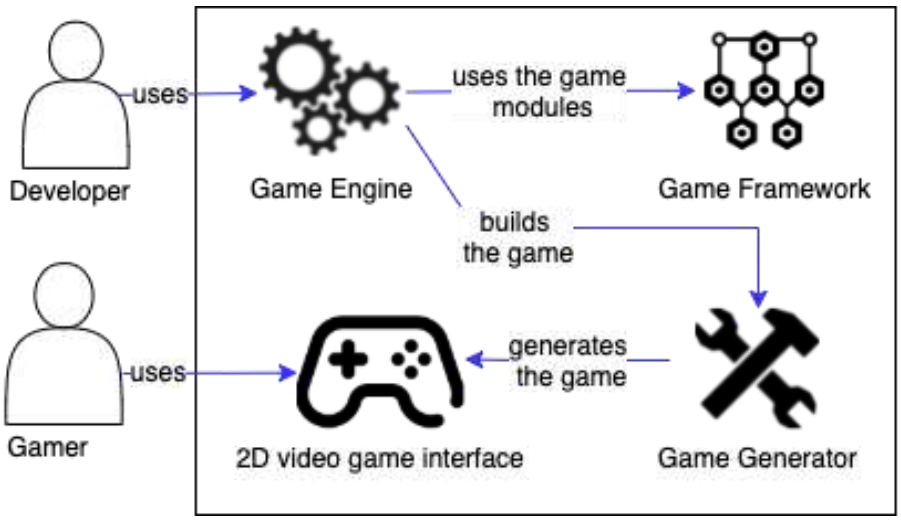
En esta sección, se presentará y detallará la arquitectura elaborada para el Game Engine y Game Framework 2D. Esta consta de dos secciones, la lógica elaborada mediante el modelo C4 y la física que se divide en dos vistas, la del desarrollador y del jugador.

4.2.1 Arquitectura Física

En esta sección se muestra la infraestructura del proyecto tanto para el jugador como para el desarrollador.

Figura 5

Perspectiva de la arquitectura física para el desarrollador y el jugador



Nota. En la figura 5, la arquitectura física muestra la interacción interna de los distintos componentes de la solución, así como la comunicación del desarrollador y jugador con el Game Engine y el juego 2D respectivamente. El Game Engine utiliza las funcionalidades del Game Framework para implementarlos y/o extenderlos, lo cual permite proveer al desarrollador de distintas capacidades y características que puede utilizar para construir su idea de juego mediante una interfaz gráfica. Finalmente, el jugador interactúa con la interfaz del juego mediante inputs como, teclado, mouse y monitor.

4.2.2 Arquitectura Lógica

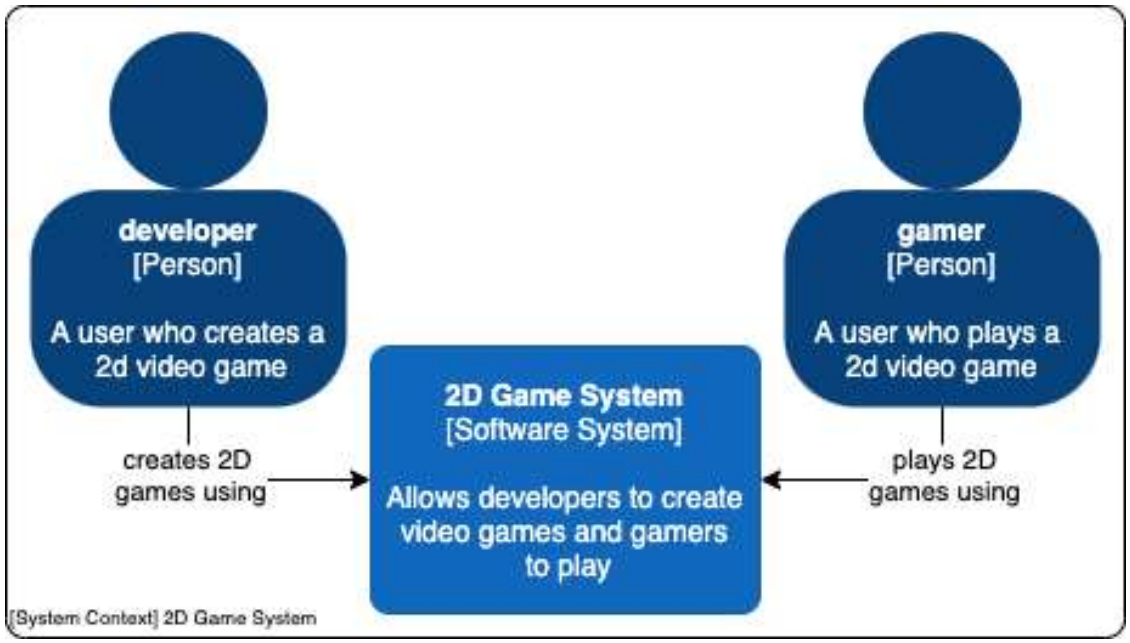
Para representar la arquitectura lógica del proyecto se utilizó el modelo C4, porque lo consideramos la forma más eficiente de ilustrar la arquitectura de un software en

diferentes niveles de abstracción, de esta manera, se puede dirigir a diversas audiencias y mostrar la información que requiera cada una de ellas.

4.2.3 Vista de Contexto del Sistema

Figura 6

Vista general del proyecto



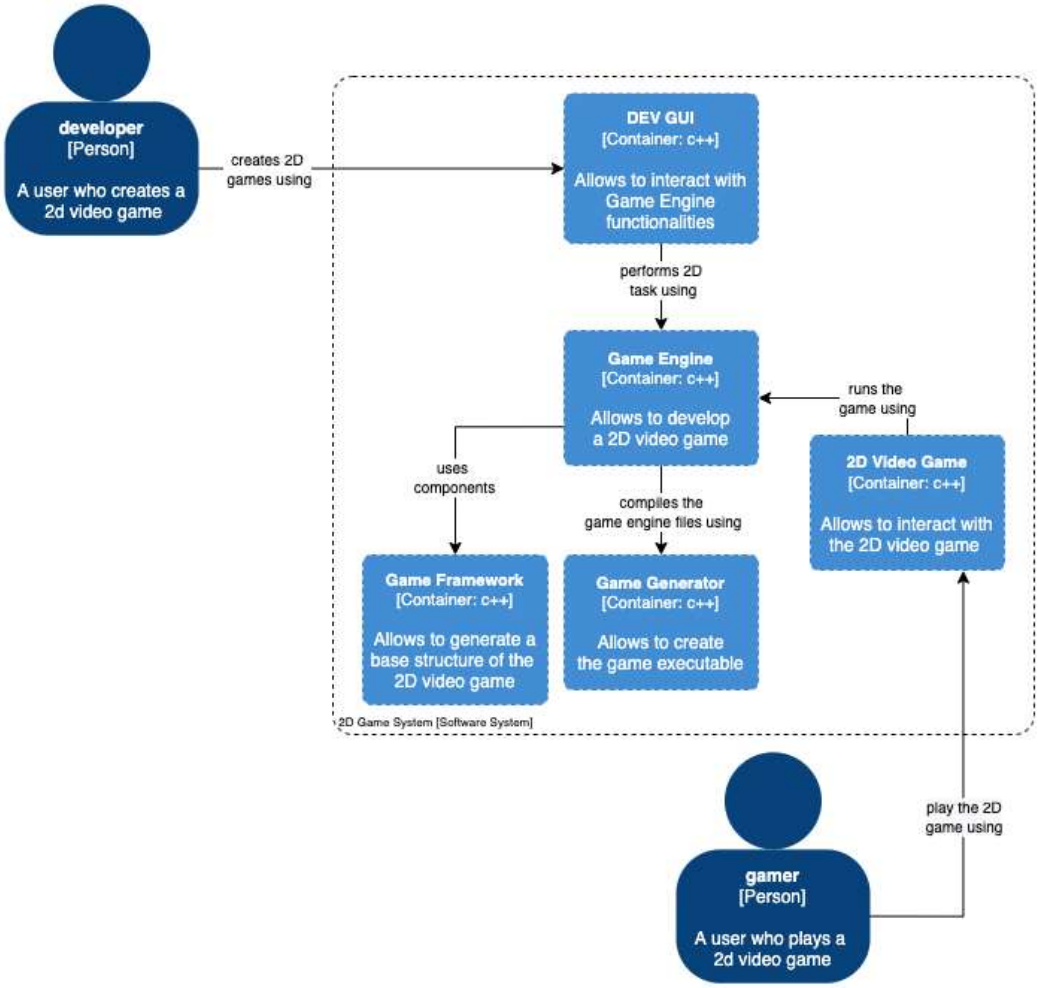
Nota. En la figura 6, se aprecia el sistema del proyecto, que cuenta con dos tipos de usuarios, el desarrollador de videojuegos y el jugador que interactúan con el sistema para construir videojuegos 2D y jugar respectivamente.

4.2.4 Vista de Contenedor

A continuación, observamos el sistema con todos los contenedores que interactúan entre sí para permite construir videojuegos 2D y jugar.

Figura *7*

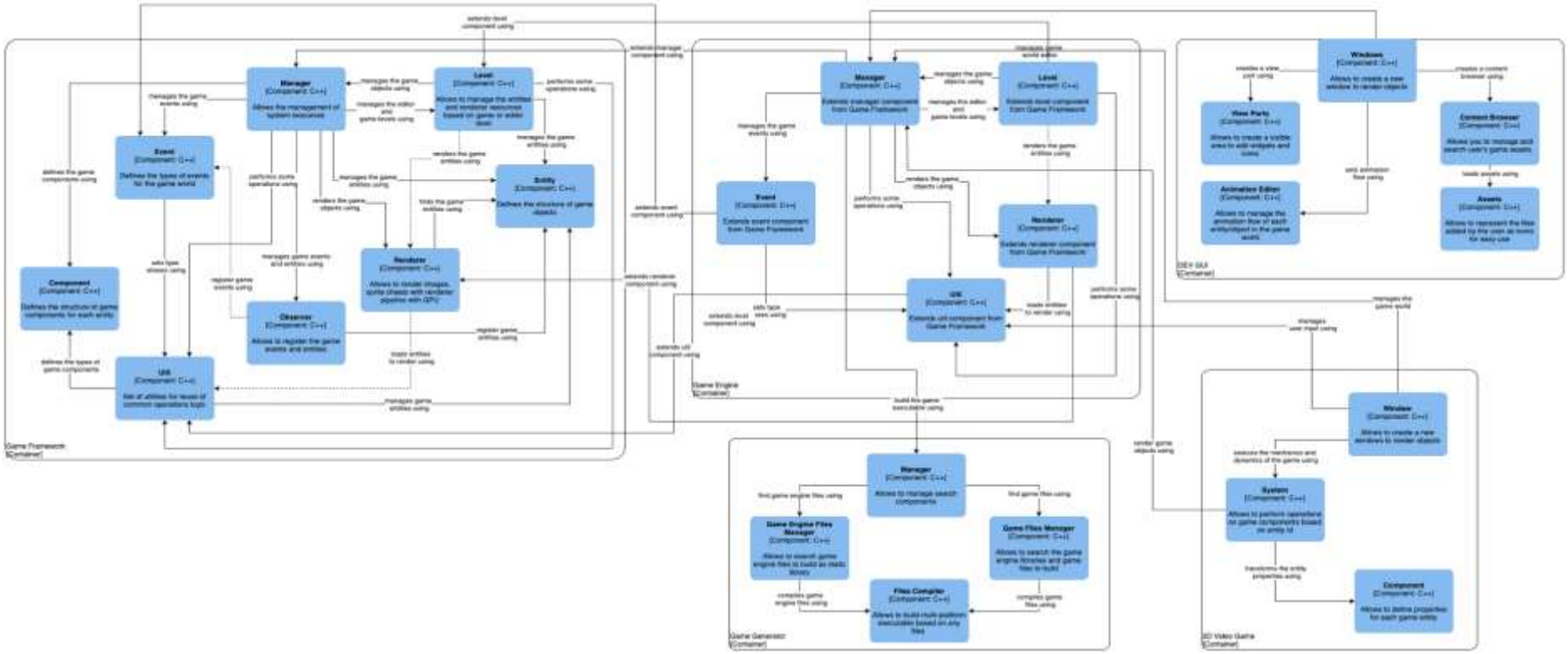
Vista de contenedores del proyecto



4.2.5 Vista de Componente

Figura 8

Vista de componentes del proyecto (Completo)



Nota. En esta sección se detalla todos los componentes que forman parte de la arquitectura lógica del proyecto.

Figura 9

Vista de componentes del proyecto (DEV GUI)

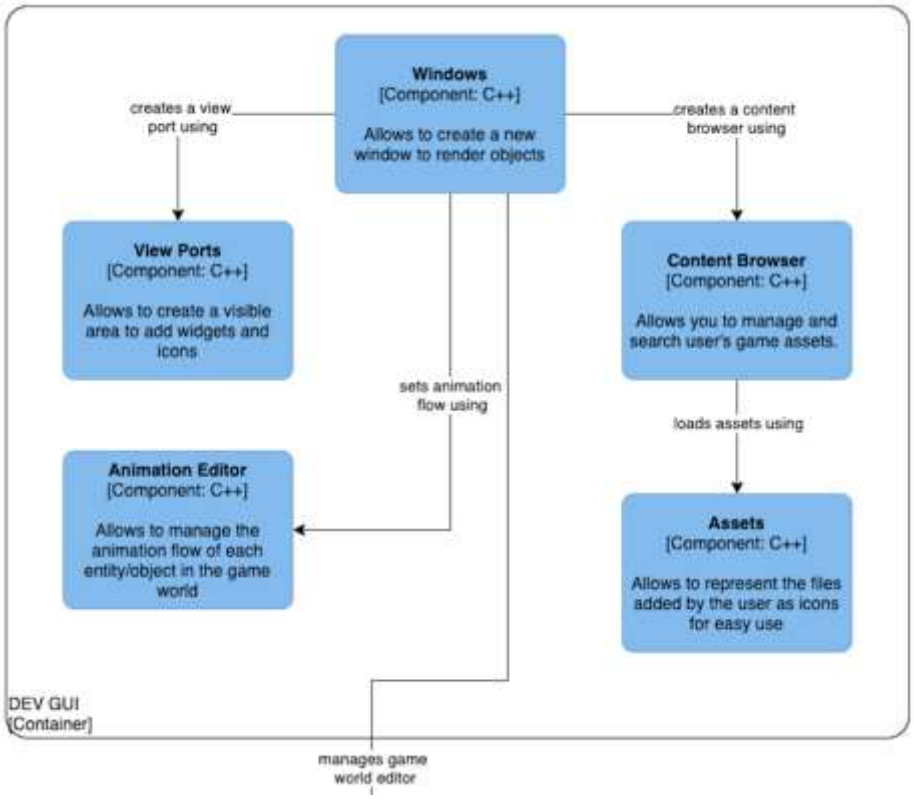


Figura 10

Vista de componentes del proyecto (Game Engine)

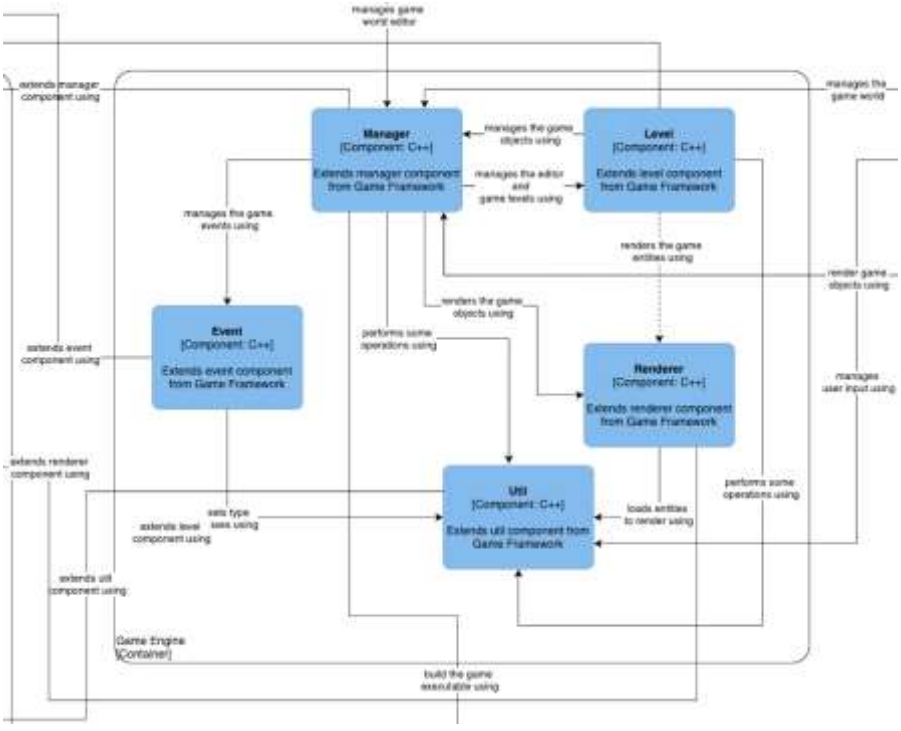


Figura 11

Vista de componentes del proyecto (Game Framework)

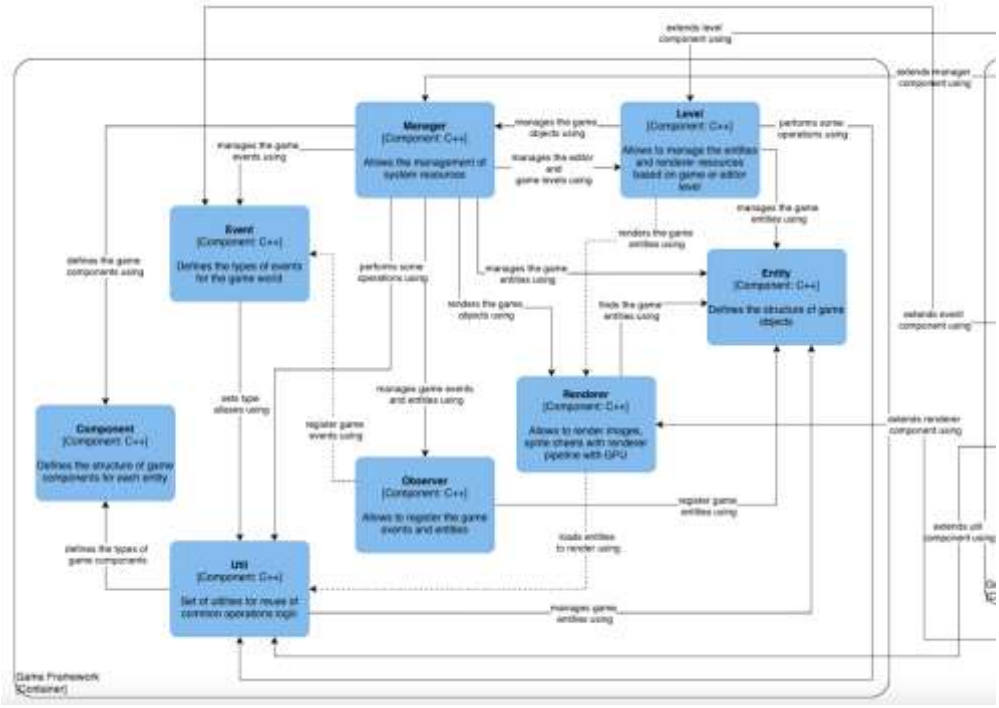


Figura 12

Vista de componentes del proyecto (2D Video Game)

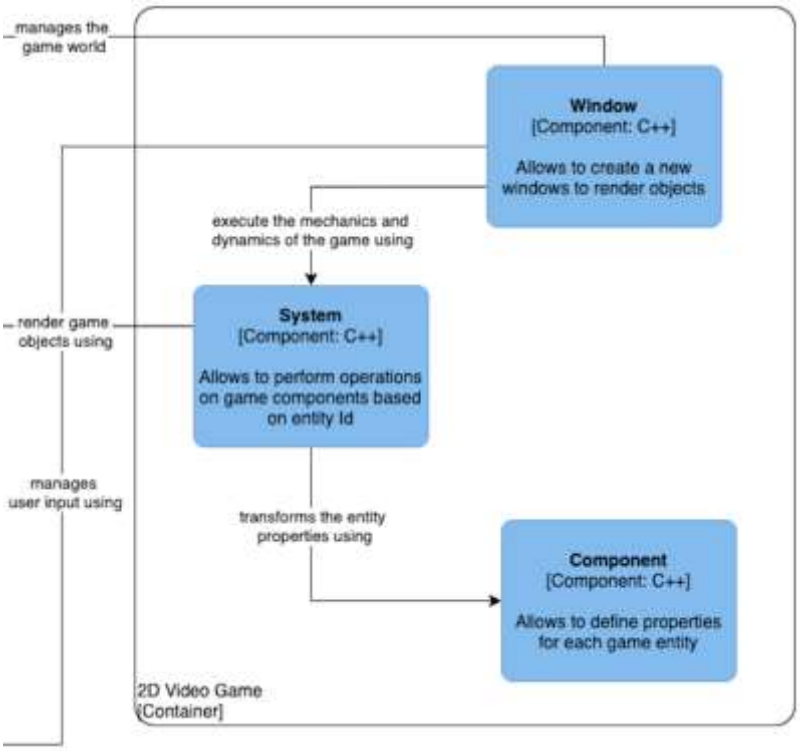
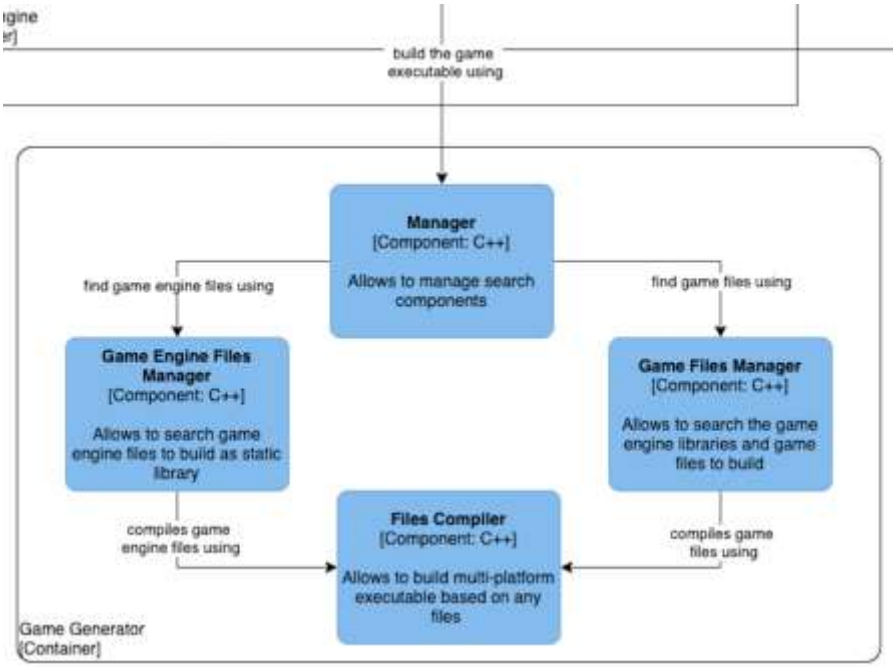


Figura 13

Vista de componentes del proyecto (Game Generator)

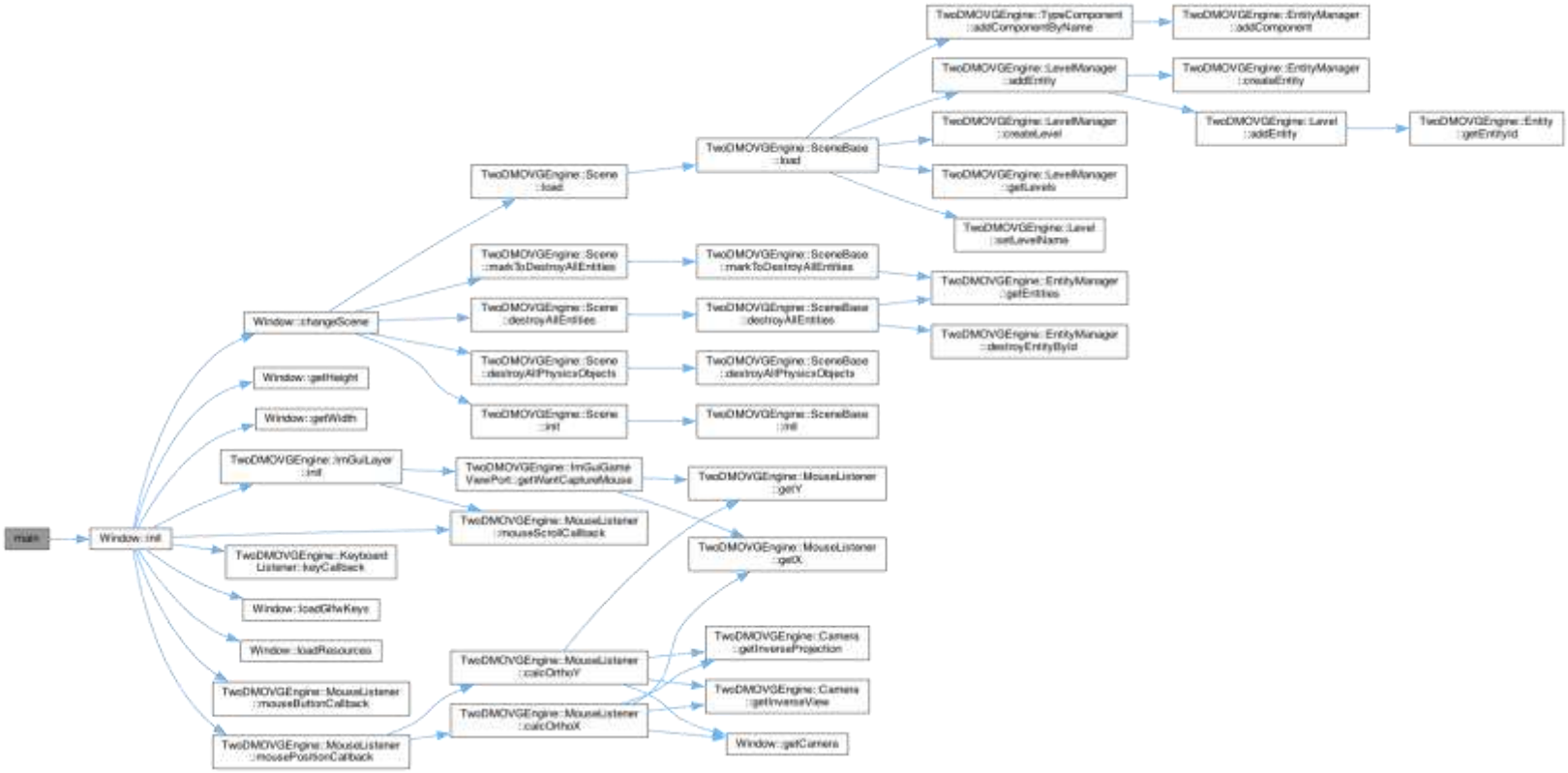


4.2.6 Vista de Código

En esta sección se detalla todos los componentes que forman parte de la arquitectura lógica del proyecto a nivel de código.

Figura 14

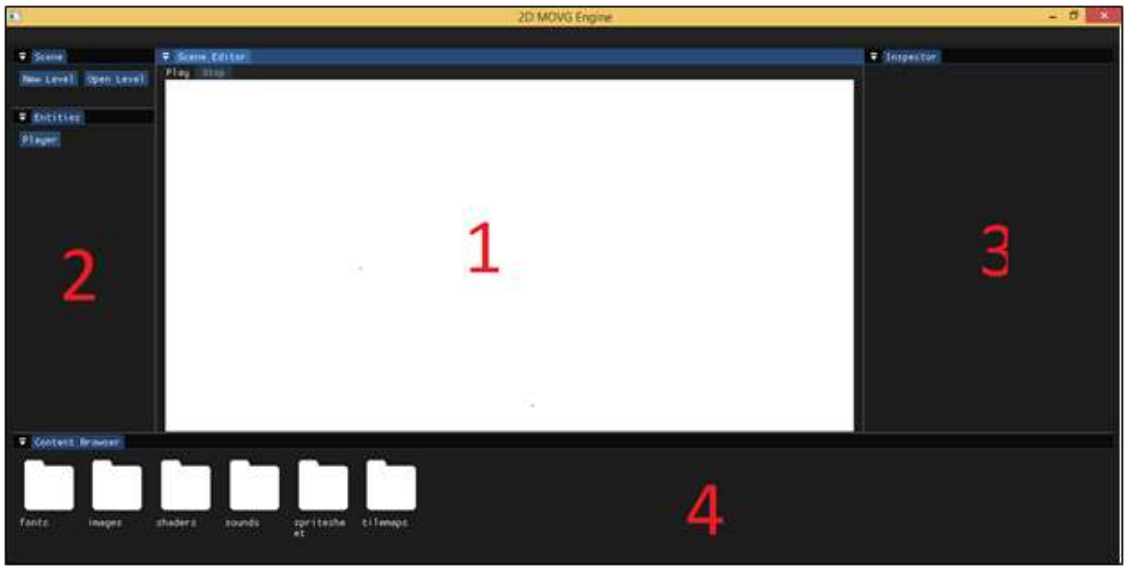
Vista de código del proyecto



4.3 Game Engine y Game Framework

Figura 15

Interfaz del Game Engine

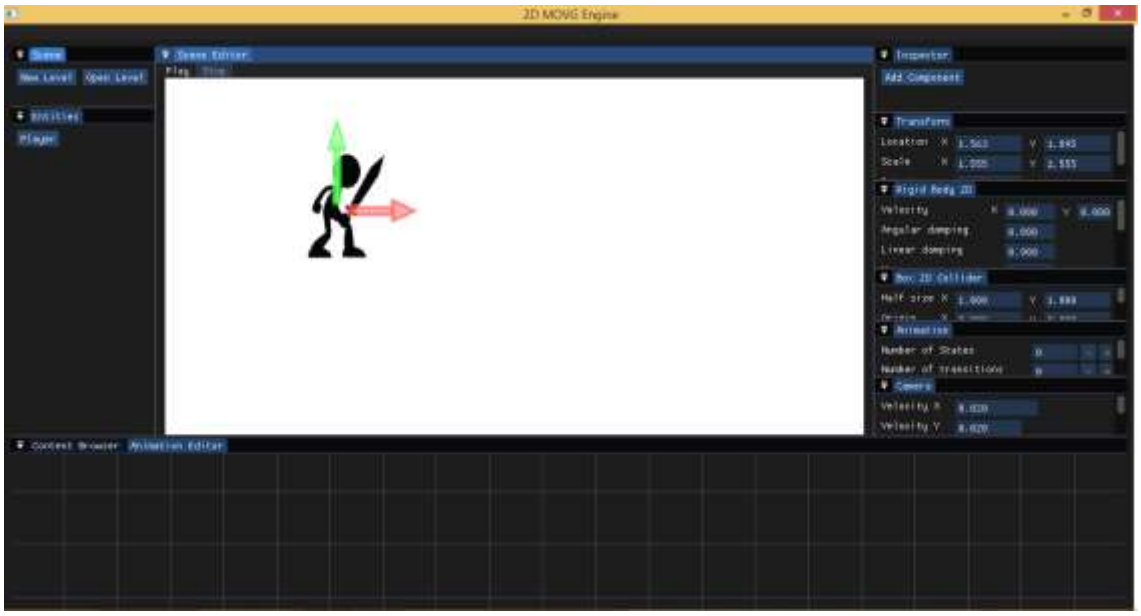


Nota. En la figura 15, se muestra la interfaz general de Game Engine con los módulos:

1. **Renderizado**: Espacio donde se renderizan los Game Objects añadidos al canvas y cumple la función de lienzo para todos los objectos.
2. **Game Objects**: Espacio con las opciones para crear entidades del tipo Player, Enemy, NPC, entre otros.
3. **Inspector de Propiedades**: Espacio donde se pueden visualizar y modificar las propiedades de los Game Objects.
4. **Administración de archivos**: Espacio donde se manejan las interacciones con el almacenamiento del dispositivo y los archivos necesarios para crear un juego, como sprites, sonidos, tile maps, entre otros.

Figura 16

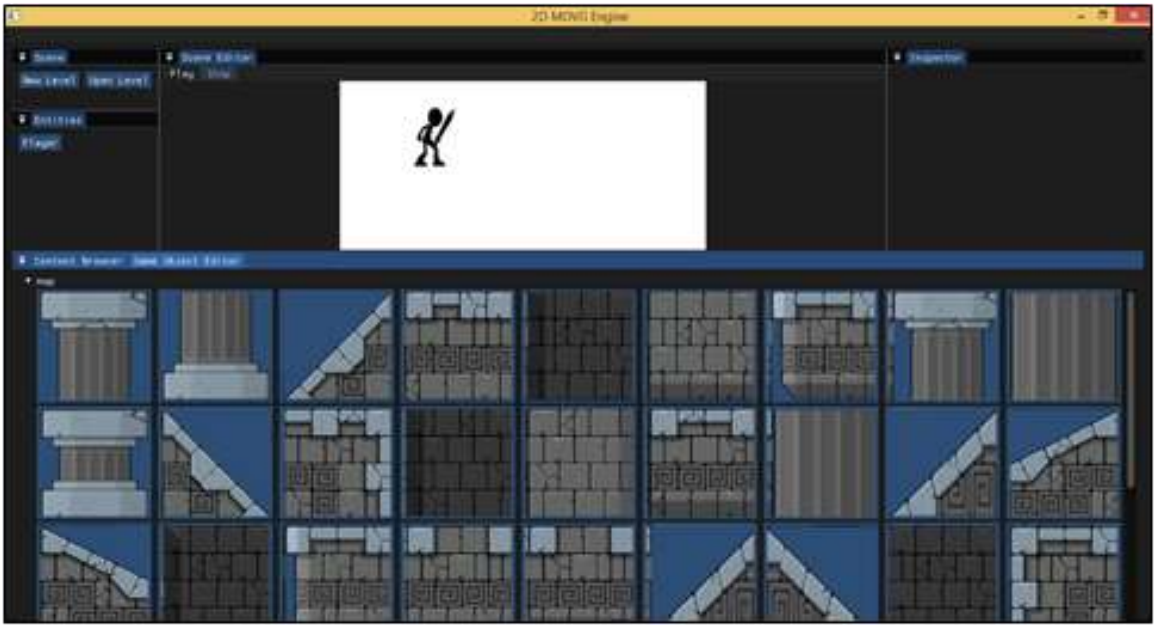
Inspector de propiedades del Game Engine & Framework



Nota. En la figura 16, se muestra las propiedades de la entidad Player y la opción de modificarlas a preferencia del desarrollador. Entre muchas de las propiedades podemos encontrar algunas como, el tamaño, ubicación, el sprite a renderizar, entre otros.

Figura 17

Content browser de los Game Objects

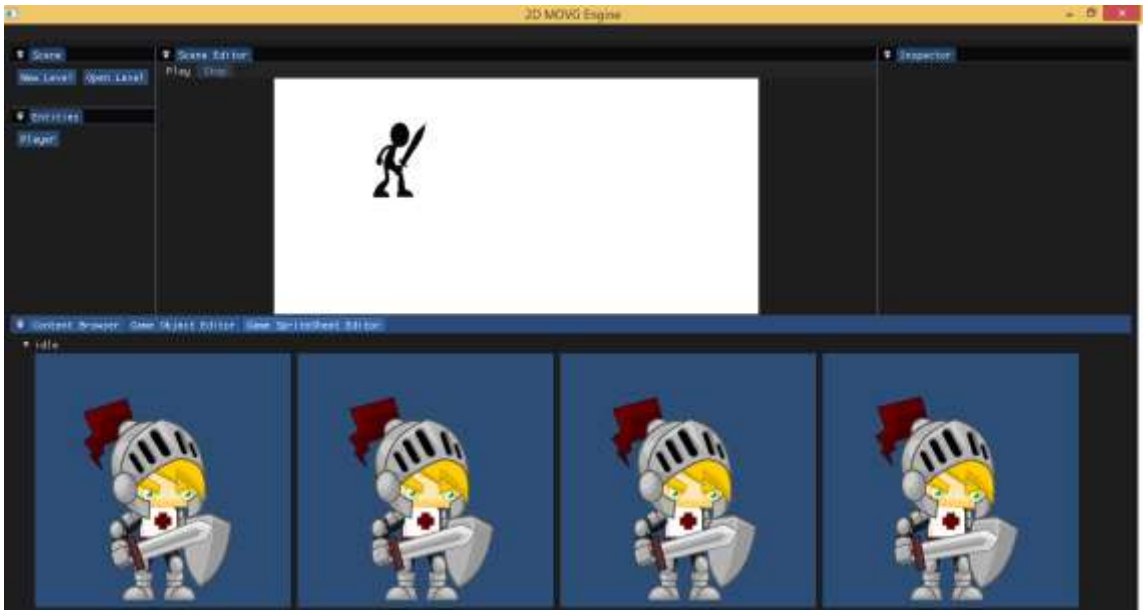


Nota. En la figura 17, se muestra cómo se cargan los tiles maps de un archivo preconfigurado para poder añadirlos simplemente arrastrando dichos elementos al canvas

del desarrollador. Al añadir un Game Object también tienen la facilidad de moverlos vertical u horizontalmente con el Gizmo.

Figura 18

Content browser del Sprite sheet



Nota. En la figura 18, se visualizan los sprites de un jugador al cargarlos desde el módulo de administración de archivos y la posibilidad de agregar las animaciones.

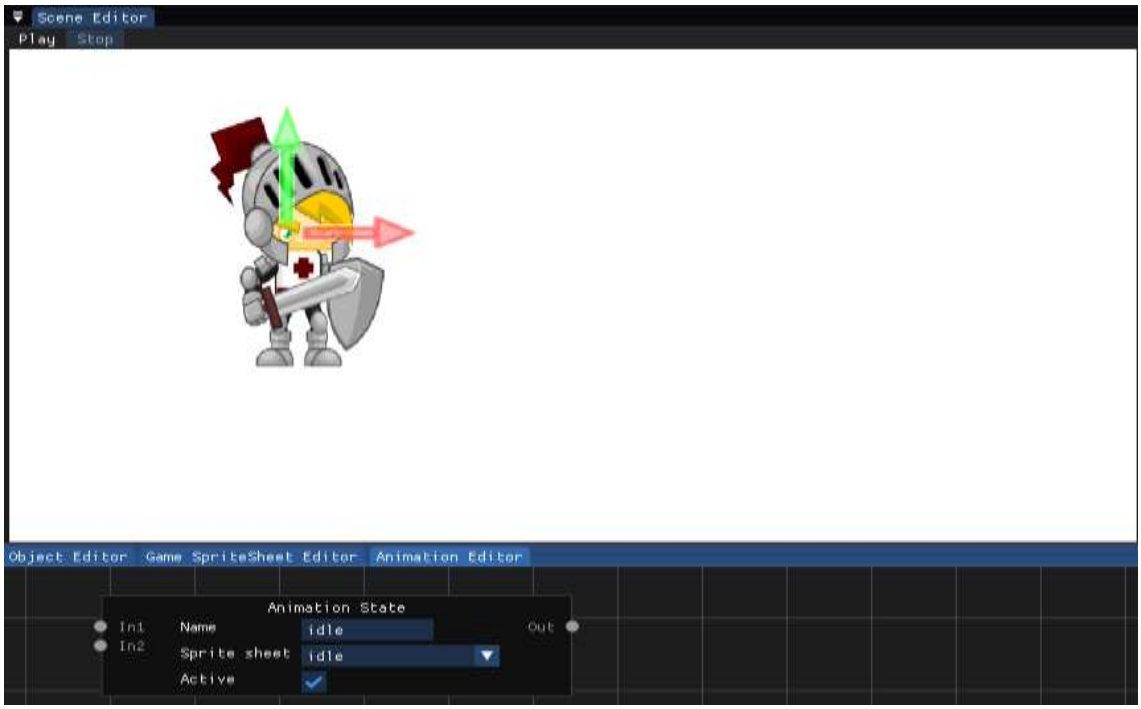
Figura 19

Gizmo en los Game Objects

Nota. En la figura 19, se visualiza las opciones de Gizmo para configurar de forma dinámica la posición y el tamaño de los Game Objects en el canvas,

Figura 20

Animation Editor para el player



Nota. En la figura 20, se visualiza las opciones de animación para los estados definidos por el desarrollador, en este caso, el estado idle para el Game Object del jugador.

Figura 21

Player y Game Objects

Nota. En la figura 21, se puede observar el Player y los Game Objects dentro de la mismo Game Scene.

5. CAPÍTULO 5: RESULTADOS DEL PROYECTO

A continuación, se detalla los resultados obtenidos al realizar la validación de la propuesta. La validación fue realizada por desarrolladores de videojuegos novatos y expertos al interactuar con el proyecto.

5.1 Pruebas de aceptación

Las pruebas de aceptación se realizaron con dos tipos de usuarios esto con la intención de contar con la opinión de novatos y expertos en el desarrollo de videojuegos.

La dinámica de validación para ambos tipos de usuarios fue que construyan un juego 2D en 27 minutos utilizando nuestro Game Engine y Game Framework. Asimismo, desarrollen dicho juego con otros programas que actualmente utilizan como, Game Engines, Frameworks, bibliotecas o programación directamente. Esto con el fin de comparar el tiempo y esfuerzo de realizar un juego 2D con nuestro proyecto y con las herramientas que manejan dichos usuarios.

A todos los usuarios se les envió un enlace con los materiales de validación vía correo electrónico. Dicho enlace contiene el ejecutable del proyecto, assets para el juego 2D, un documento sobre los requisitos del juego, un manual de uso y un video tutorial donde se realiza el juego propuesto en 27 minutos. El video demo puede encontrarse en el siguiente enlace: https://upcedupe-my.sharepoint.com/:v:/g/personal/u201520167_upc_edu_pe/Eb b6Vxg1yxFDtQR41bfMHYUBk0o8FD6pPPmEvwz41lPLvA?e=rGNJi5

5.1.1 Validación con novatos

La validación fue realizada con una muestra de 10 desarrolladores novatos que construyen juegos por diversión o por algún curso en la universidad. Para esta actividad, se realizó una presentación por video llamada a los 10 usuarios juntos, donde se les explicó el proyecto y se les mostró una demo del Game Engine.

5.1.2 Validación con expertos

La validación fue realizada con una muestra de 2 desarrolladores expertos que construyen juegos como parte de su trabajo diario. Para esta actividad, se realizó una presentación por video llamada a cada uno, donde se les explicó el proyecto y se les mostró una demo del Game Engine.

5.2 Aplicación de encuesta

5.2.1 Cuestionario

Una semana después de entregarles la demo a los participantes, se les envío un formulario para que nos brindaran su opinión con respecto al proyecto, este cuestionario tiene como finalidad mostrar la apreciación de los usuarios al interactuar con nuestro proyecto, para ello se consideraron los siguientes criterios, la experiencia de usuario y la utilidad de todas las funcionalidades en el flujo de desarrollo de juegos 2D.

El cuestionario fue realizado y aplicado según las regulaciones de la LPDP (Ley de Protección de Datos Personales), donde:

- Se envió un cuestionario vía correo electrónico a los usuarios explicándoles el motivo (evaluar el proyecto).
- Únicamente se les solicitó nombres y apellidos, y correo electrónico (datos personales).

Los datos personales se trataron o analizaron siempre manteniendo la misma finalidad informada a los encuestados previamente vía correo electrónico, como lo ordena la LPDP (Congreso de la República del Perú, 2011, Ley 29733, Artículo 6). Además, los usuarios dieron su consentimiento expreso de utilizar sus datos personales al dar click al botón enviar del formulario, tal como se menciona la LPDP (Congreso de la República del Perú, 2011, Ley 29733, Artículo 12).

Las preguntas realizadas fueron puntuadas de 1 a 5 con la escala de Likert, ya que usar un rango de 1 a 3 permitiría la presencia de ambigüedad en las respuestas tal como mencionan Joshi et al. (2015).

Tabla 7

Niveles de escala según el tipo de respuesta

Tipo de respuesta	Escala
Totalmente de acuerdo	5
De acuerdo	4

Ni de acuerdo ni en desacuerdo	3
En desacuerdo	2
Totalmente en desacuerdo	1

El cuestionario se divide en 4 secciones:

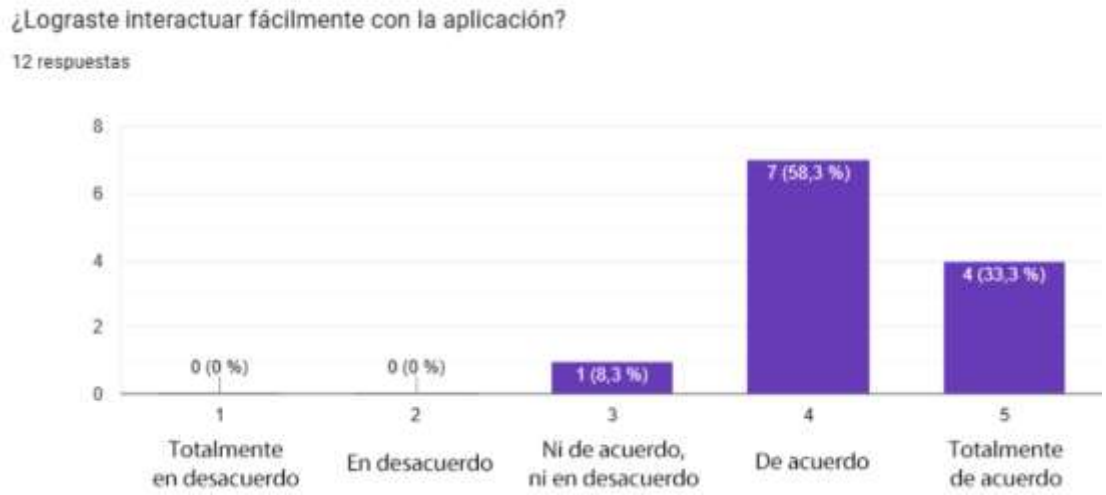
1. Información básica del participante (Nombre completo, correo, experiencia desarrollando videojuegos, entre otros) – 4 preguntas
2. Experiencia de usuario en general (Sin compararla con otros Game Engine) – 2 preguntas
3. Experiencia de usuario de las funcionalidades comparadas con otros Game Engine – 13 preguntas
4. Opiniones finales del Game Engine - 1 pregunta

El formulario puede ser revisado en el siguiente enlace: https://forms.gle/HsJ9ecdPgh14XCPV8.

Interacción

Figura 22

Resultados de interacción

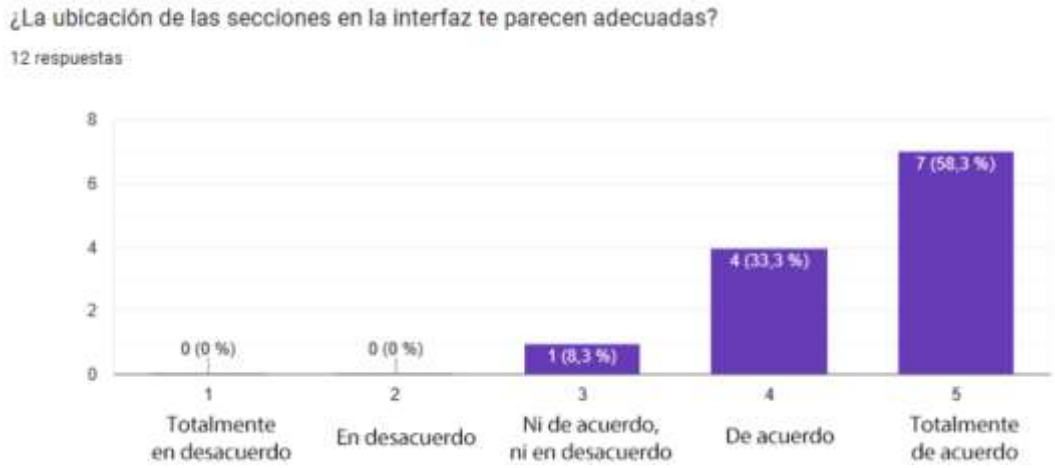


Nota. En la figura 22, observamos que el 91.6% les pareció sencillo o muy sencillo la interacción con el proyecto, en específico, el 33.3% de usuarios estuvieron totalmente de acuerdo que la interfaz contiene las opciones necesarias para el desarrollo de juegos 2D.

Diseño

Figura 23

Resultados del diseño

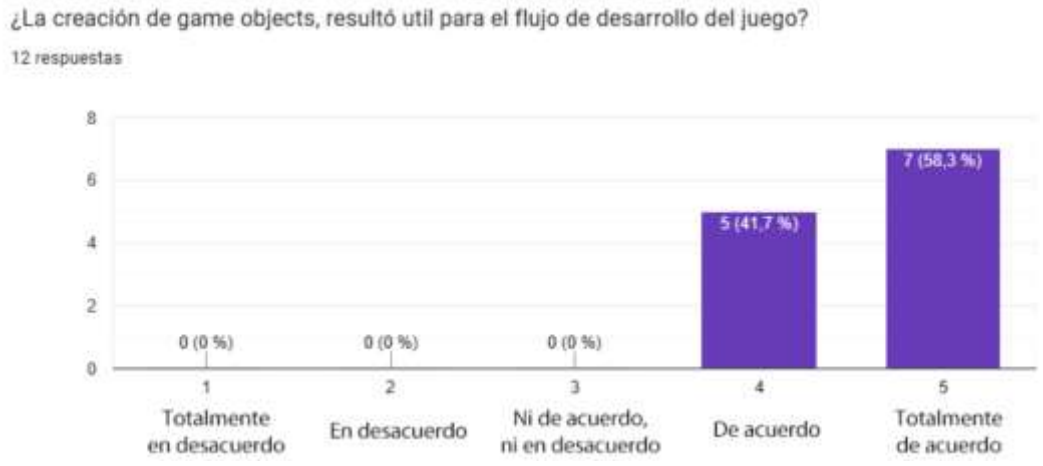


Nota. En la figura 23, muestra que el 91.6% de usuarios les pareció correcta la ubicación de cada una de las opciones del Game Engine. Asimismo, el 8.3% de los encuestados no estuve ni a favor ni en contra con la ubicación de cada sección del proyecto.

Creación de Game Objects

Figura 24

Resultados de la creación de Game Objects

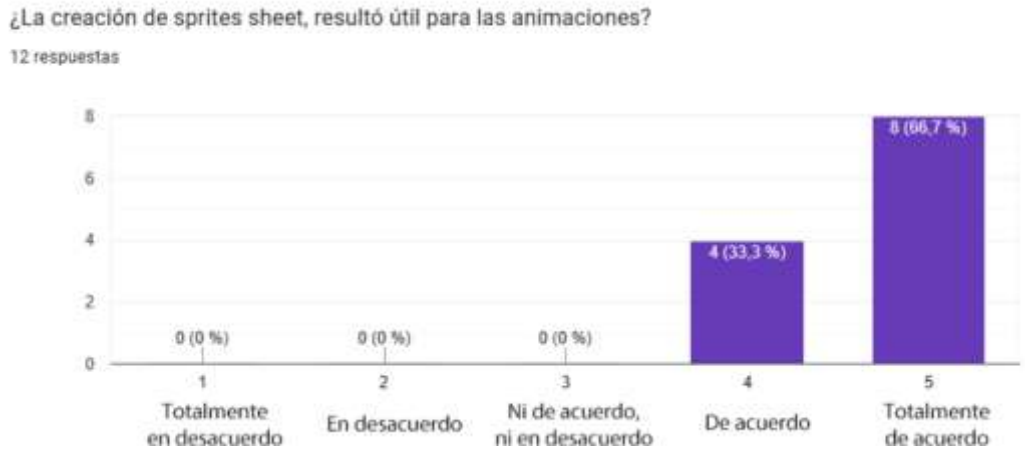


Nota. En la figura 24, se evidencia que al 100% de los encuestados estuvieron de acuerdo o totalmente de acuerdo en que la creación de game objects facilita el flujo de desarrollo del videojuego, ya que brinda componentes precargados específicos para personajes y para objectos que pueden ser personalizados.

Creación de Sprite sheets

Figura 25

Resultados de la creación de Sprite sheets

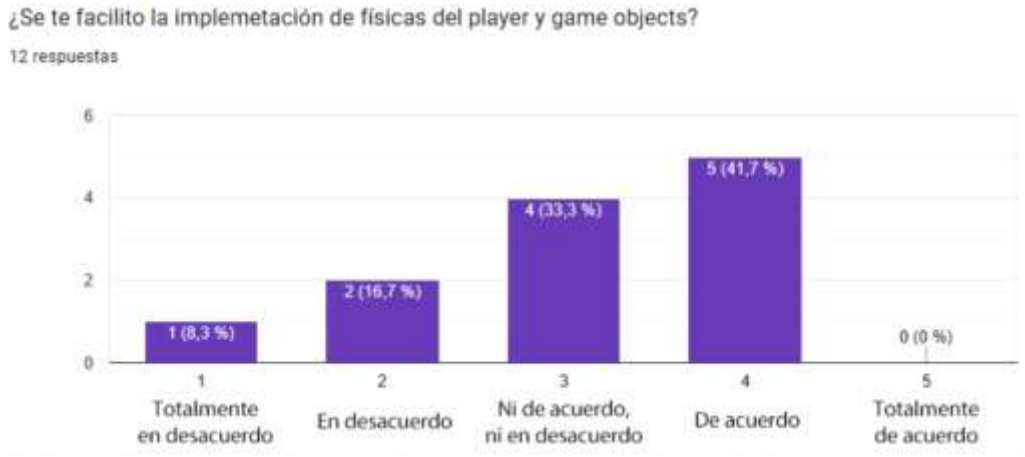


Nota. En la figura 25, se evidencia que al 66.7% de los desarrolladores estuvieron totalmente de acuerdo que la creación de sprite sheet, ayuda en el proceso de animación de player y game objects, por tanto, simplifica el tiempo del proceso al realizarlo de forma manual. Asimismo, el 33.3% estuvo de acuerdo que dicha funcionalidad es útil en el proceso de animación.

Física del juego

Figura 26

Resultados de la física del juego



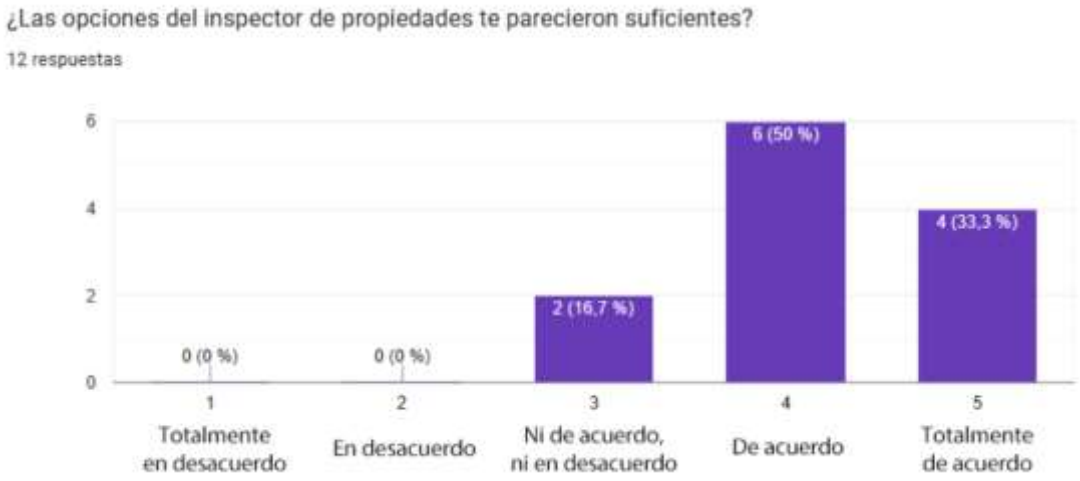
Nota. En la figura 26, se evidencia al 41.7% de encuestados les resultó útil la configuración de física en el desarrollo del videojuego. Asimismo, al 33.3% no les facilito

ni les complicó en el flujo de desarrollo del videojuego. Sin embargo, al 25% les pareció muy difícil o difícil utilizar dicha funcionalidad, lo cual indica que se debe de realizar mejoras para lograr una interacción más eficiente con los usuarios.

Inspector de componentes

Figura 27

Resultados del inspector de componentes

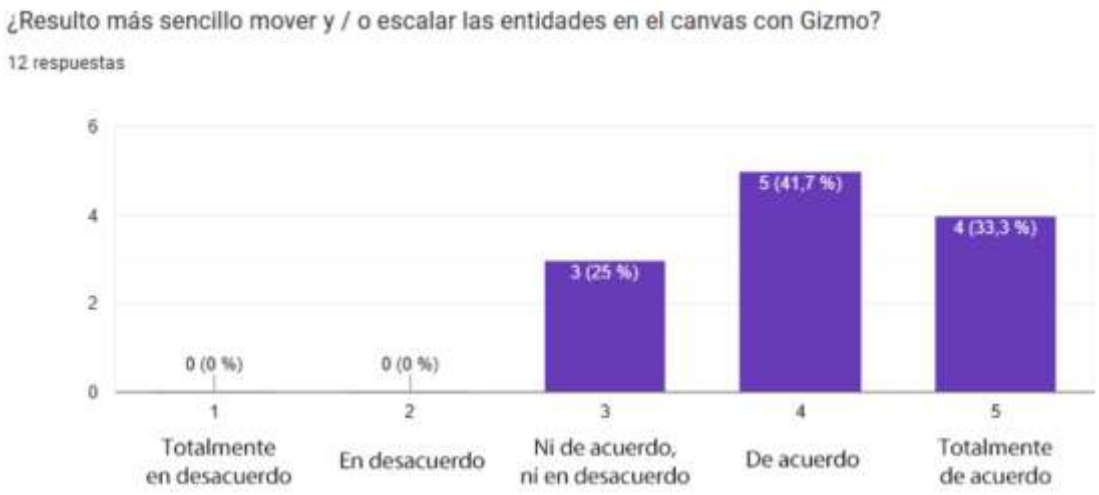


Nota. En la figura 27, notamos que el 83.3% de los usuarios estuvieron totalmente de acuerdo o de acuerdo con la cantidad de propiedades del inspector de componentes, mientras que solo el 16.7% no votaron ni a favor ni en contra.

Gizmo

Figura 28

Resultados de Gizmo

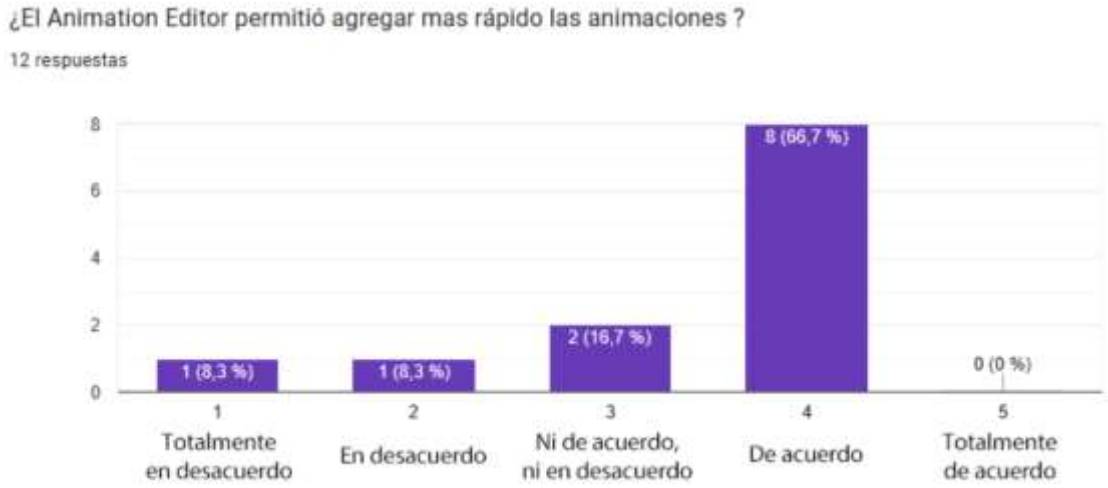


Nota. En la figura 28, observamos que al 75% de los encuestados les pareció muy sencillo o sencillo mover y/o escalar las entidades en el canvas 2D al momento de desarrollar el videojuego 2D. Además, al 25% nos les favoreció ni afectó dicha funcionalidad en su flujo de desarrollo.

Animation Editor

Figura 29

Resultados del Animation Editor

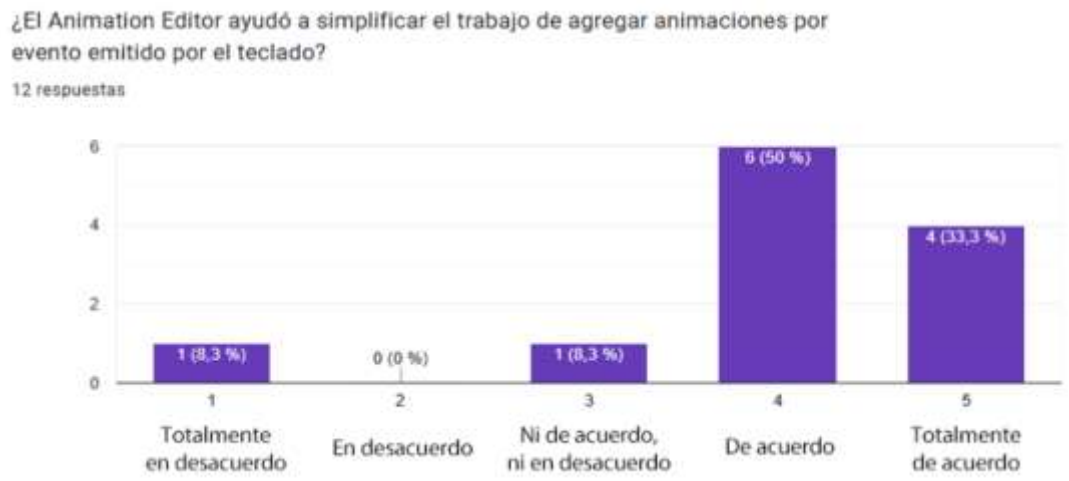


Nota. En la figura 29, observamos que al 66.7% de encuestados les ayudó la característica de Animation Editor para la animación de los estados del player. Sin embargo, al 33.3% les ayudo en poco o nada en simplificar las animaciones del player y de los game objects. Esto indica que debemos trabajar en mejorar las características e interacción.

Animaciones por evento

Figura 30

Resultados de las animaciones por evento

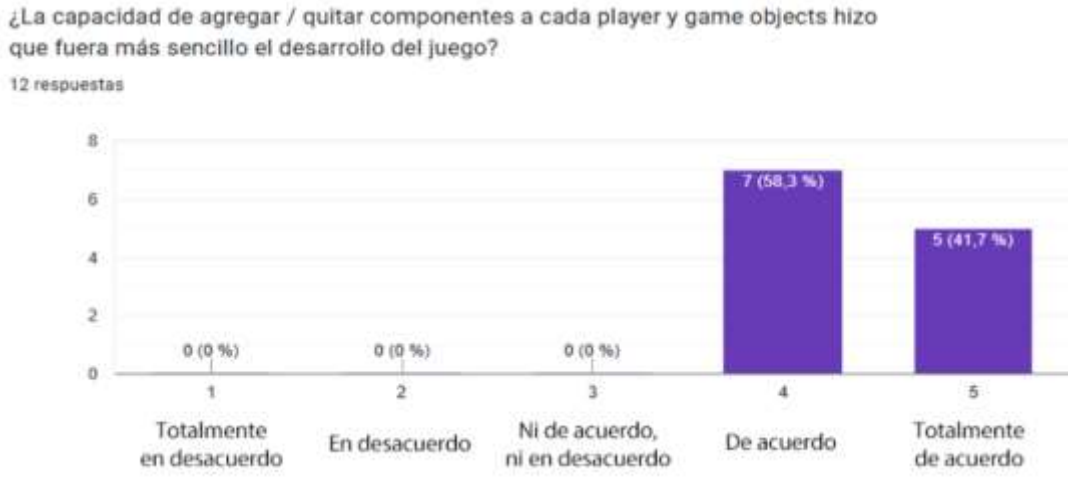


Nota. En la figura 30, se evidencia que al 16.6% de los encuestados les ayudo muy poco o poco el Animation Editor en el proceso de animación por eventos del player y de los game objects. Sin embargo, al 83.3% simplificaron dicha tarea con la funcionalidad implementada.

Editor de componentes

Figura 31

Resultados del editor de componentes

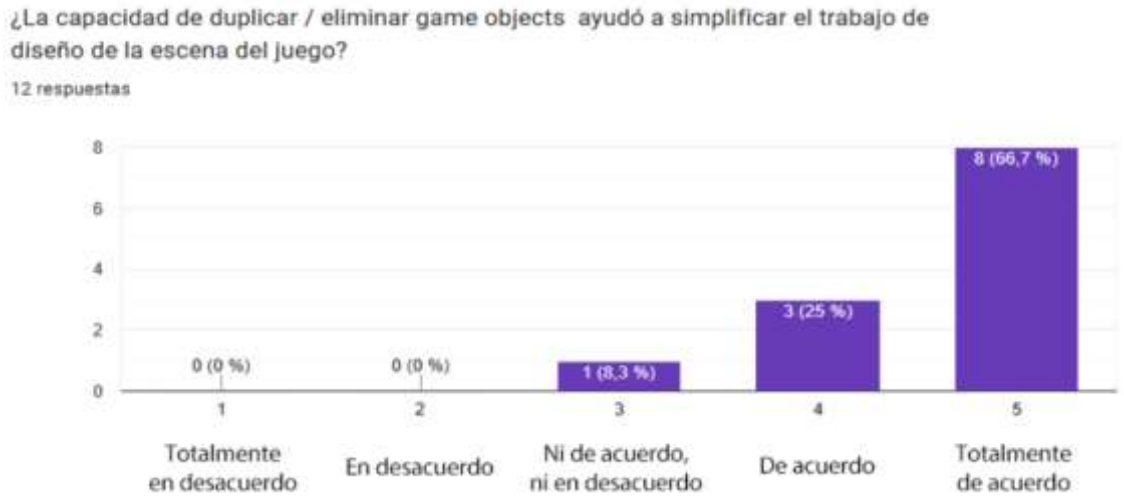


Nota. En la figura 31, se observa que el 41.7% de los encuestados les resultó de mucha utilidad la configuración de componentes para el desarrollo del videojuego, mientras que al 58.3% restante, les pareció útil dicha funcionalidad en la construcción del juego.

Manipulación de Game Objects

Figura 32

Resultados de la manipulación de Game Objects



Nota. En la figura 32, se observa que el 91.7% de encuestados lograron simplificar el trabajo de diseño de la escena del juego usando la opción de duplicar y eliminar game objects.

Cámara del juego

Figura 33

Resultados de la cámara del juego

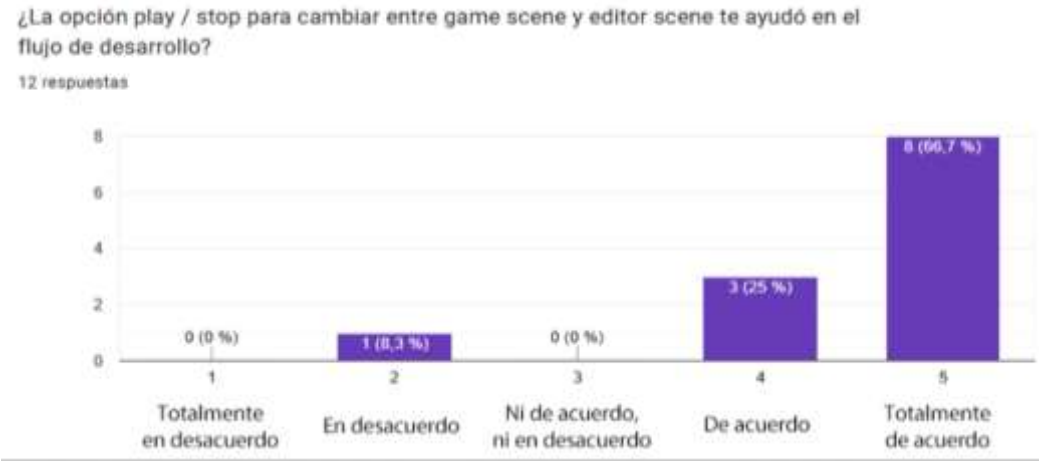
Nota. En la figura 33, se observa que al 25% de los encuestados les fue indiferente la capacidad de mover la cámara y el zoom frente a otras herramientas. Asimismo, el 75%

restante les ayudó en el proceso de desarrollo del videojuego 2D. Esto nos indica que podríamos mejorar más esta funcionalidad para que esa útil a todos los usuarios.

Game & Editor Scene

Figura 34

Resultados de Game & Editor Scene

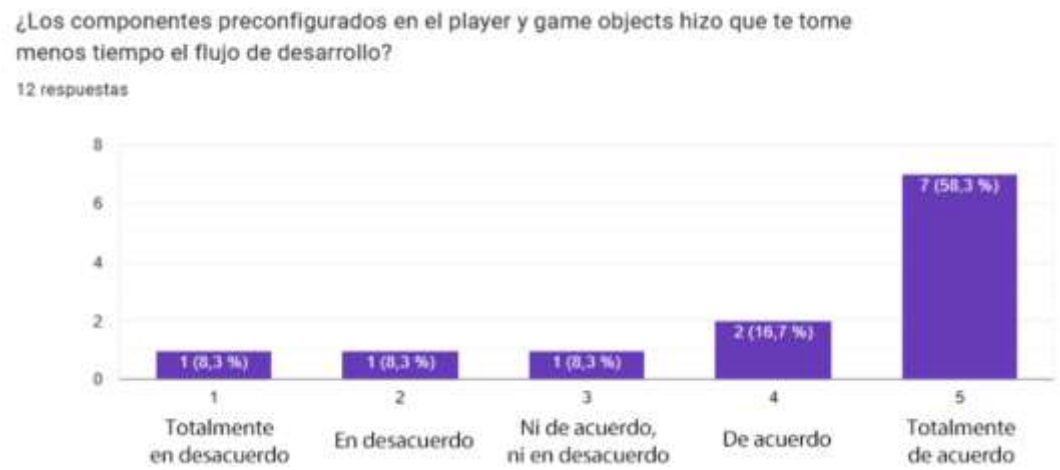


Nota. En la figura 34, se observa que al 8.3% de los encuestados no les resultó útil la capacidad de intercambiar el modo de edición entre Game Scene y Scene Editor durante la construcción del juego. Sin embargo, al 91.7% les ayudó la funcionalidad mencionada en el flujo de construcción del videojuego.

Componentes preconfigurados

Figura 35

Resultados de los componentes preconfigurados

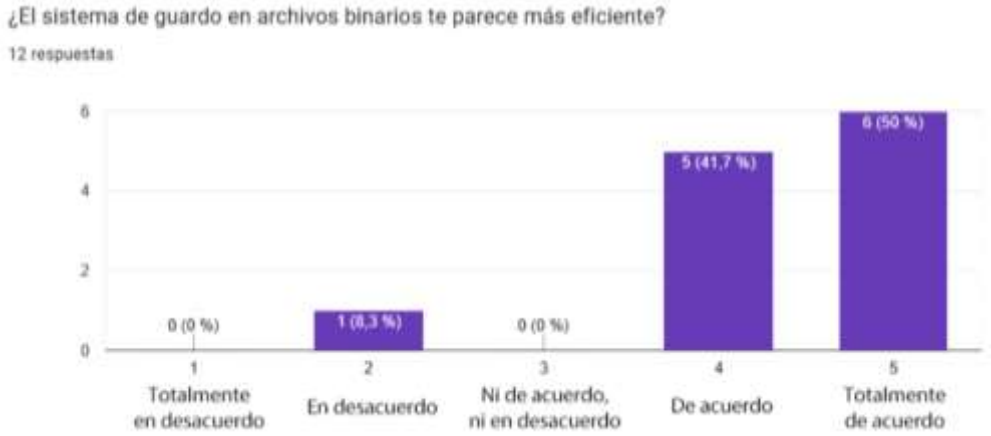


Nota. En la figura 35, como se puede ver los componentes preconfigurados fueron útiles para el 75% de desarrolladores de juegos. Por otra parte, al 25% les ayudo en poco o nada, lo cual indica que aún debe ser mejorada para obtener mejores resultados.

Archivo de guardado de datos

Figura 36

Resultados del archivo de guardado de datos



Nota. En la figura 36, se observa que al 8.3% de los encuestados no les pareció apropiado el formato binario para guardar los archivos del juego. Pero, al 91.7% les pareció muy eficiente o eficiente el uso del formato binario en el guardado de datos del progreso de desarrollo del videojuego.

Añadir proyecto

Figura 37

Resultados sobre añadir el proyecto al set de herramientas

Nota. En la figura 37, se observa que solo el 25% de los encuestados aún no están convencidos en incluir nuestro Game Engine y Game Framework en su conjunto de herramientas.

5.3 Análisis de resultados

Como se aprecia en los gráficos, las dos primeras preguntas se refieren a la experiencia de usuario. Donde observamos lo siguiente:

- El 96.1% de los usuarios interactuaron de forma sencilla o muy sencilla con la interfaz del Game Engine, dado que solo contiene opciones para entornos 2D. Es decir, no satura de alternativas innecesarias al desarrollador como, el manejo de esferas, figuras abstractas en 3D, configuración de esqueletos, adición de materiales, animación tridimensional, entre otros, que están orientados a la tecnología 3D.
- El 96.1% de los encuestados estuvo de acuerdo con las ubicaciones de cada una de las opciones del proyecto, por tanto, podemos decir que el diseño de la interfaz de usuario ha sido exitoso.

Finalmente, la mayoría de los usuarios votaron entre 4 y 5, salvo un par de casos que votaron en 3. Esto significa que la mayor parte de los usuarios lograron interactuar exitosamente con nuestro proyecto, lo cual indica que la interfaz de usuario tiene un diseño eficaz y usable.

Las siguientes preguntas están enfocadas en las funcionalidades que impactan en el flujo de desarrollo de un juego 2D, tanto en el esfuerzo de programarlos o configurarlos y el tiempo que toma realizarlos.

- En el caso de la creación de Game Objects y Sprite sheets, los votos estuvieron en un puntaje entre 4 y 5, lo cual indica que les ayudó a los usuarios en realizar dichas tareas, en lugar de realizarlas a mano.
- En el caso de la física del juego fue una funcionalidad que solo ayudó al 41.7% de usuarios, debido a la dificultad de configuración, esto indica que se tiene que mejorar la interacción y reducir su complejidad.

- En el caso del inspector de componentes, ayudó en el proceso de configuración de propiedades del player y los game objects, según el 83.3% de usuarios. Sin embargo, el 16.7% votó de forma neutra, esto puede deberse a que actualmente el panel de configuración no cuenta con un diseño adaptable o "responsive design", lo cual dificulta un poco la interacción a medida que aumentan los componentes. Por tanto, debe ser un punto de mejora en una próxima versión del proyecto.
- En cuanto al uso de Gizmo, según el 75% de usuarios les resultó útil contar con esta funcionalidad, ya que permite mejorar el manejo de la ubicación y la configuración de la gematría de las entidades. Sin embargo, al 25% les fue indiferente esta característica, esto puede deberse a que según aumenta la cantidad de game objects o de players, resulta un poco más complicado mover los ejes de Gizmo. Por tanto, debe ser considerado como un punto a mejorar como parte de la experiencia de usuario.
- En cuanto al editor de animaciones, tuvo un 66.7% de aceptación por parte de los usuarios. Pero, el 33.3% les ayudó en poco o nada al momento de añadir y/o configurar las animaciones. Por tanto, se debe mejorar el canvas de animación para mejorar la interacción con las cajas de estados y la generación de conexiones entre animaciones.
- Al 100% de los encuestados les pareció muy útil o útil el editor de componentes, ya que permite añadir de forma dinámica nuevas propiedades y configurarlas, lo cual simplifica la tarea de programar características de forma manual.
- La opción de duplicar y eliminar Game Objects para facilitar el diseño de escenarios ayudó al 91.7% de los usuarios, lo cual indica que fue exitoso la implementación de dicha funcionalidad, pero aún puede ser mejorado.
- El uso de la cámara para el proceso de construcción del videojuego ayudó al 75% de los usuarios. En contraste, al 25% le fue indiferente esta funcionalidad. Esto nos permite entender que aún debemos trabajar en mejorar el movimiento de la cámara para una mejor experiencia y facilidad de uso.
- La funcionalidad de Game & Editor Scene ayudó al 91.7% de los encuestados, dado que simplifica la tarea de cambiar de modo de desarrollo a modo de prueba sin necesidad de compilar a cada momento el juego.
- El 75% de los usuarios les resultó muy útil o útil los componentes preconfigurados en el flujo de desarrollo del videojuego. Pero, al 25% no le fue de utilidad, esto

puede deberse a que se han agregado muchos componentes precargados a cada tipo de entidad. Por tanto, se debe evaluar los más importantes para ser añadidos por defecto.

- El guardado de datos en formato binario fue exitoso, ya que el 91.7% de los desarrolladores les pareció adecuado este formato de archivo.
- En cuanto a incluir nuestro proyecto en su lista de herramientas, el 25% de los encuestados tienen dudas, en parte puede deberse a que el proyecto aún tiene varios puntos a mejorar y todavía no está listo para producción.

Finalmente, la mayoría de los usuarios votaron entre 4 y 5, salvos algunos casos en los que votaron en 3 o menos. Esto indica que nuestro proyecto en general ayudó a disminuir el esfuerzo y el tiempo en construir juegos 2D, que al utilizar otros Game Engines genéricos, Framework, bibliotecas o programación desde 0. Asimismo, aún necesita realizar mejoras.

6. CAPÍTULO 6: GESTIÓN DEL PROYECTO

A continuación, se detalla las tácticas, actividades y herramientas utilizadas para gestionar el proyecto.

6.1 Plan de gestión del Proyecto

6.1.1 Fases del proyecto

Tabla 8

Fases del proyecto

Fase	Revisiones	Criterio de entrada	Criterio de salida
Inicio	Se realizará la revisión del Project Charter Benchmarking de los Game Engine y Game Framework	Documento de Project Charter Documento de Benchmarking	Documento de Project Charter aprobado por el PM. Benchmarking aprobados por el PO.
Planificación	Se revisará el documento de "Gestión de alcance". Se revisará el documento de Plan de gestión de cronograma Se revisará el documento de Gestión de calidad Se revisará el documento de Gestión de riesgos Se revisará el documento de Gestión de comunicaciones	Documentos de Gestión de proyectos. Wireframes del Game Engine. Suit de Pruebas	Documentos de "Gestión de Proyectos" aprobados por el PM. Wireframes aprobados por el PO

	Se revisará el documento de Plan de gestión de requisitos Se revisará el documento de Plan de gestión de costos Se revisará el documento de Gestión de adquisición Se revisará el documento de Plan de gestión de recursos Wireframes del Game Engine Plan de pruebas		
Ejecución	Se revisará el desarrollo del Game Framework Se revisará el desarrollo del Game Engine Se revisarán los objetivos específicos 2 y 3	Avance del Game Framework. Avance del Game Engine. Documento de los objetivos específicos.	Documento de objetivos específicos aprobados.
Cierre	Acta de cierre de proyecto. Memoria Anexo WASC Paper aprobado	Acta de cierre de proyecto Documento WASC Documentos de Memoria	Aprobación del acta de cierre de proyecto. Paper final aprobado.

		Paper aprobado	

6.1.2 Enfoques de desarrollo

Tabla 9

Enfoques de desarrollo

Entregable	Enfoque de desarrollo
Project Charter	Adaptativo
Documento de Students Outcome	Adaptativo
Artefactos de gestión de proyectos	Adaptativo
Documentos de análisis	Adaptativo
Documento de Product Backlog	Adaptativo
Documento de arquitectura	Adaptativo
Prototipos de la aplicación	Adaptativo
Documento de planes de prueba	Adaptativo
Documento de memoria de proyecto.	Adaptativo
Acta de cierre de proyecto	Adaptativo

6.1.3 Planes de Gestión Subsidiaria

Tabla 10

Planes de gestión subsidiaria

Nombre	Comentario
Alcance	El proyecto estará de la mano con lo requerido por el Product Owner. Por lo tanto, habrá una constante comunicación con este stakeholder para recibir un constante feedback y de esta manera, poder tener un alcance fijo.
Tiempo	El proyecto constará con 29 semanas de trabajo lo cual implica dos semestres académicos.
Costo	Los costos están ligados a las herramientas que vamos a utilizar y el recurso humano que serán los programadores. Estos costos serán cubiertos en su totalidad por los alumnos.
Calidad	La calidad del proyecto será asegurada por los certificados QS.

Recurso	La estimación de los recursos necesarios será detallada en el documento de Plan de Gestión de los recursos.
Comunicaciones	Este punto será detallado en el documento de Plan de gestión de Comunicaciones.
Riesgo	Este punto será detallado en el documento de Plan de gestión de Riesgos
Logro	El logro del proyecto estará ligado al cumplimiento de los objetivos, alcance, tiempo, costos y calidad establecidos.
Stakeholder	Los Stakeholders del proyecto son el Portafolio Manager, Product Owner y el comité de proyectos.

6.1.4 Umbral de Variación de Alcance

La modificación del alcance inicial del proyecto implica un riesgo tanto en costos como en las fechas de entrega, ya que puede ocasionar cambios en el proyecto a nivel de documentos, artefactos, actividades, dependencias, entre otros (Juárez, 2016). Por tanto, cualquier cambio que se realice al alcance del proyecto debe ser revisada y aprobada por los Stakeholders del proyecto.

6.1.5 Gestión del Alcance de la Línea Base

El proyecto tiene como finalidad construir un Game Engine que utilizará su propio Game Framework para facilitar el desarrollo de videojuegos 2D. Además, dicho alcance fue aprobado por el PM y PO. Finalmente, cualquier cambio a realizarse debe pasar por un control de cambios para disminuir los riesgos del proyecto (Juárez, 2016).

6.1.6 Umbral de Variación del Calendario

Villavicencio (2020) señala que alterar el cronograma de trabajo genera riesgos en el proyecto a nivel de entrega y costos. Asimismo, indica que para cumplir con las tareas planificadas dentro del calendario se debe utilizar el método Crashing para incrementar los recursos sin afectar el tiempo del proyecto. Por ello, se tendrá un límite máximo de 5% en la modificación de las fechas de finalización de las actividades para no generar costes altos.

6.1.7 Gestión de Línea de Base del calendario

Las modificaciones al calendario inicial deben de ser realizadas mediante un control de cambios y aprobados por el PM y PO (Villavicencio, 2020). Asimismo, dicha línea inicial del cronograma puede ser encontrado en el Plan de trabajo del proyecto.

6.1.8 Umbral de Variación de Costos

El coste del proyecto puede incrementar solo hasta un 5% por encima de la línea base. Además, si el aumento es superior a lo establecido se tiene que evaluar acciones a realizar para asegurar el cumplimiento con el presupuesto definido (La guía PMBOK, 2013).

6.1.9 Gestión de la línea Base de los Costos

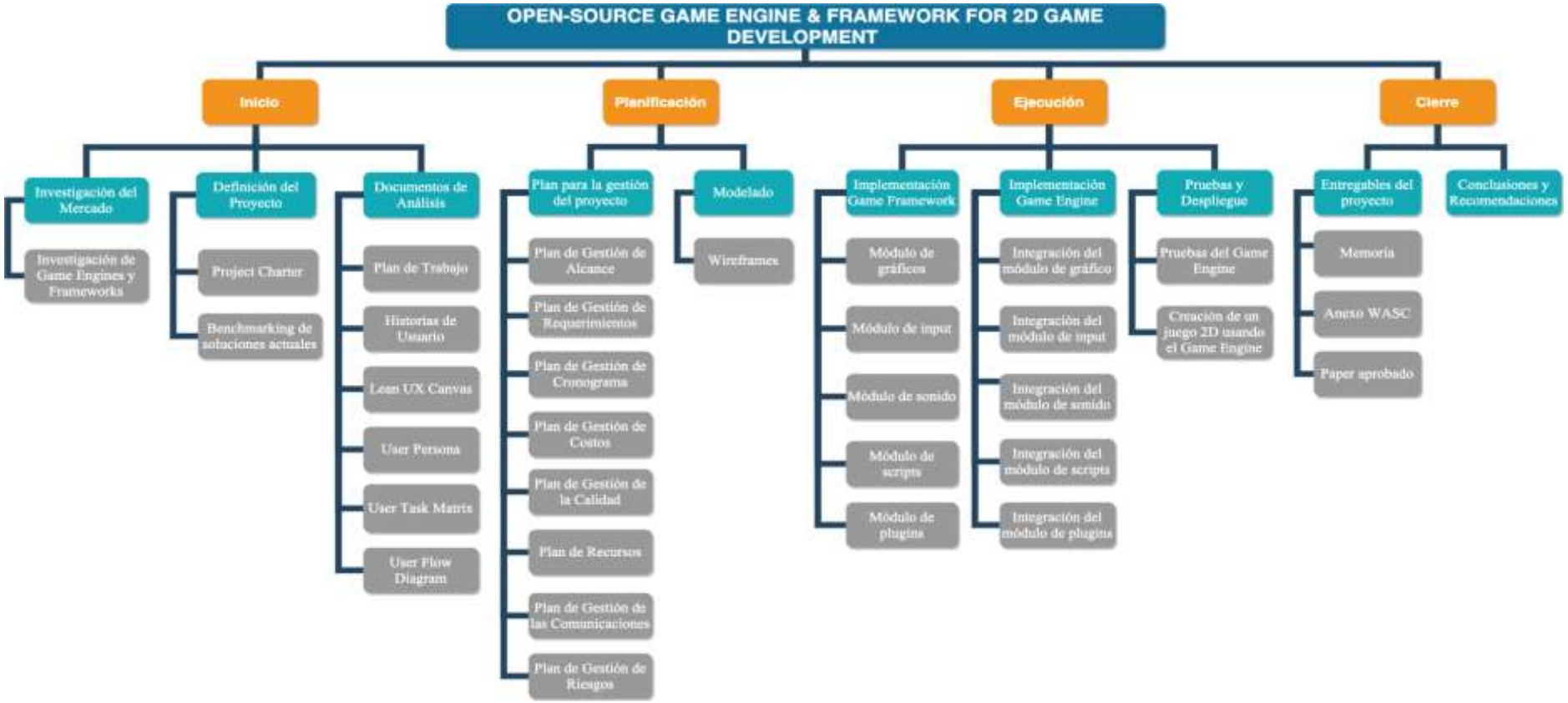
El coste inicial para el presente proyecto es el que fue aprobado por el PM y PO para proseguir con el desarrollo de este (Gascón, 2017). Asimismo, el detalle de los costos inicial puede ser revisado en la tabla 16 del anexo c.

6.2 Plan de gestión de alcance

6.2.1 EDT

Figura 38

EDT



6.2.2 Diccionario EDT

- Project Charter: Documento donde se presentan los objetivos y la importancia del proyecto, así como los requisitos de alto nivel, riesgos, alcance e interesados.
- Plan de Gestión del Alcance: Documento que cuenta con el enunciado del alcance del proyecto y la EDT.
- Plan de Gestión de Requisitos: Documento que contiene los requisitos del producto y la relación con los entregables correspondientes.
- Plan de Gestión de Cronograma: Documento que contiene distintas fases como definición, secuencia y duración de las actividades, así como la gestión y control del cronograma.
- Plan de Gestión de Costos: Documento en donde se definen los costos estimados por cada actividad y las reservas de contingencia.
- Plan de Gestión de Calidad: Definición de las métricas de calidad del proyecto.
- Plan de Recursos: Documento en donde se definen los recursos humanos del proyecto y los paquetes a los que son responsables cada uno. Asimismo, se elabora un calendario de los recursos para saber en qué periodos trabajarán.
- Plan de Gestión de las Comunicaciones: Documento donde se refleja la estrategia de comunicación con los interesados.
- Plan de Gestión de Riesgos: Documento en donde se identifican los riesgos del proyecto, la realización del análisis cualitativo y cuantitativo; y las respuestas ante estos.
- Lean UX Canvas: Documento donde se formulan hipótesis comprobables y planes de experimentación en base a una serie de suposiciones.
- User Persona: Documento que contiene la representación idealizada de un cliente ideal, la cual está basada en los estudios de un mercado potencial.
- User Task Matrix: Documento donde se identifican las principales acciones de cada User Persona en base a importancia y veces que se realizan.
- User Flow Diagram: Diagrama que representa todos los pasos que el usuario debe realizar hasta lograr el objetivo final.
- Wireframes: Bocetos donde se representa visualmente y de forma muy sencilla la interfaz y estructura de un producto.
- Benchmarking: Documento que contienen las comparaciones de los diferentes productos actuales con los que competirá la solución.

- Plan de pruebas: Documento que contiene todos los casos de prueba a realizar en el sistema desarrollado. Considera las pruebas de caja blanca, caja negra, funcionales y no funcionales.

6.2.3 Alcance de la línea base de Mantenimiento

Dado un cambio que modifique la línea base del proyecto, se realizará una solicitud de cambio al comité y este tendrá que aprobarlo. En caso de que el cambio no afecte la línea base del proyecto, solo deberá ser aprobado por el jefe del proyecto.

6.2.4 Aceptación de Entregable

Los entregables realizados deben ser con total aprobación de nuestro PO/Cliente. La validación del alcance del proyecto se realizada con el Portafolio Manager. Y finalmente en cada Sprint se realizarán informes de desempeño en función a cada entregable aprobado.

6.2.5 Alcance e Integración de requisitos

Para desarrollar la EDT el equipo se basó en el plan de trabajo inicial y los requisitos del proyecto. Mientras que para la planificación se hizo uso diversos elementos como el Lean UX Canvas, planes de gestión del proyecto y creación de bocetos.

6.3 Plan de gestión de requisitos

6.3.1 Recolección de Requisitos

- Historias de Usuario: Documento que describe las necesidades de los usuarios del proyecto y los criterios de aceptación.
- User Lean UX: Un enfoque de diseño que permite construir de forma ágil los productos de software.
- User Persona: Personajes imaginarios que permiten representar los usuarios del producto de software a utilizar.

6.3.2 Análisis de Requisitos

Los requisitos serán analizados mediante:

- User Task Matrix: El cual permite clasificar las funcionalidades más relevantes por tipo de usuario.

- Clasificación de Historias por categoría, usuario y técnico. En este caso son prioritario los del tipo técnico, ya que tienen que ver con el desarrollo del código fuente.

6.3.3 Categorías de Requisitos

Los requisitos se clasifican por dos tipos de usuario:

- Desarrollador del juego: El cual utiliza la interfaz de usuario del Game Engine para construir videojuegos 2D.
- Jugador: El cual utiliza el videojuego 2D para divertirse.

6.3.4 Documentación de Requisitos

Los requisitos son documentados mediante:

- Formato Historias de Usuario. Contiene un formato que expresa claramente las necesidades del usuario y los criterios de aceptación.
- Canvas de Lean UX: En el cual se describe la ruta que debe seguir el proyecto para cumplir con todos los requisitos.
- Descripción del User Persona: Representación imaginaria de los usuarios de nuestro proyecto, el desarrollador y autor del juego.

6.3.5 Priorización de Requisitos

A continuación, se muestra la priorización de requisitos:

- Los que están relacionado con el diseño e implementación del Game Engine y Game Framework.
- Los que están relacionados con la generación de binarios de tipo biblioteca y de ejecución, ya que son necesarios para enlazar el Game Framework con el Game Engine y exportar el videojuego respectivamente.
- Los que están relacionados con la interfaz del Game Engine, esto porque el proyecto debe contar con una forma eficaz y visual de exponer sus funcionalidades.

6.3.6 Métricas

Las métricas que se usan para medir los requisitos del proyecto son:

- El Game Engine permitirá desarrollar videojuegos 2D con mayor eficiencia en tiempo y esfuerzo comparado con otros Game Engines, Frameworks, bibliotecas y programación desde cero.
- La curva de aprendizaje de las opciones del Game Engine debe ser más baja en comparación a los productos del mercado actual.

6.3.7 Estructura de Trazabilidad

La estructura de trazabilidad será realizada mediante una matriz, el cual describe los requisitos, los objetivos del proyecto, la prioridad, los usuarios y los criterios de aceptación.

6.3.8 Seguimiento de Requisitos

Para realizar el seguimiento de requisitos se utilizará:

- Microsoft Planner, monitorear el estado de cada Historia de Usuario tales como, To Do, In Progress, To be Test, Tested y Done

6.3.9 Requisitos de Reporting

Los requisitos son reportados mediante:

- Sprint Review al Product Owner, al iniciar cada Sprint

6.3.10 Validación de Requisitos

Los requisitos serán validados mediante:

- La aprobación del Product Owner sobre las Historias de Usuario
- Certificaciones:
    - Historias de Usuario
    - Arquitectura física y lógica
- Prueba del Game Engine que utiliza su propio Game Framework, mediante desarrolladores novatos y expertos.

6.3.11 Gestión de la Configuración de Requisitos

- Revisión del Product Owner de los entregables.
- Número de entregables terminados dentro del plazo.
- Número de entregables terminados fuera del plazo.

6.4 Plan de gestión de cronograma

6.4.1 Metodología de Asignación del Cronograma

Para llevar a cabo este proyecto se utilizará la metodología PMBOK, que permite la planificación del cronograma. Dicha metodología nos ayuda en establecer distintas fases para terminar el proyecto de forma segura. Faces como, definición, secuencia y duración de las actividades, así como la gestión y control del cronograma.

6.4.2 Herramientas de Planificación y Asignación del Cronograma

Las herramientas que se utilizaran son

- Microsoft Excel: Para la creación y actualización del diagrama de Gantt

6.4.3 Nivel de Exactitud

El diagrama de Gantt para este proyecto estará compuesto de plazos de finalización de tareas por semana. Es decir, se medirá el avance de las actividades del proyecto semanalmente, lo cual coincide con las unidades de medida.

6.4.4 Unidades de Medida

Las unidades para medir la realización de las actividades son:

- Sprint de 3 semanas
- 3 Daily Scrum por semana para determinar las actividades a realizar

6.4.5 Umbrales de Variación

Dado a que el avance de actividades se realiza semana a semana mediante los daily scrum y sprint review cada 3 semanas, entonces los umbrales de variación son:

- Todas las actividades planeadas para cada semana, en principio deben ser completadas, en caso de ser culminadas antes del tiempo, se realizará un ajuste del cronograma para adelantar actividades.
- En caso de haber retrasos en la culminación de actividades se ajustarán las actividades para superponerlos en la siguiente semana.

6.4.6 Asignación de Informes y Formatos

Los informes y formatos son:

- Sprint Planning en formato excel presentado al Project Manager (PM)
- Sprint Review presentado al Product Owner (PO) en formato PPT
- Actas de reunión con el Product Owner (PO)

6.4.7 Enlaces de Procedimientos Organizacionales

Para definir el cronograma de actividades se usará el EDT, el cual nos permite gestionar el alcance del proyecto.
Mediante el EDT se definirá el trabajo/componentes que entregará proyecto, esto nos ayuda en la generación del cronograma de actividades para hacer posible la entrega de componentes.

6.4.8 Actualización del Cronograma

Para realizar modificaciones al cronograma establecido, se debe sustentar al Product Owner (PO) y Project Manager (PM), luego de su aprobación, se debe transcribir la solicitud de cambios de manera formal. Asimismo, el Project Manager (PM) es responsable de enviar la solicitud con el documento redactado por los miembros del proyecto.

6.5 Lista de Hitos

Tabla 11

Lista de hitos

Hito	Descripción del Hito	Tipo
Hito 1: Validación del Project Charter	Validación del Project Charter con el Product Owner y el Portfolio Manager.	Obligatorio
Hito 2: Sustentación con el Portfolio Manager	Sustentación parcial de los artefactos de PM, el Marco teórico, objetivo 1 al 100% y Estado del Arte.	Obligatorio

Hito 3: Sustentación parcial	Sustentación de los entregables de los hitos anteriores y el avance del objetivo 2, y los atributos de calidad.	Obligatorio
Hito 4: Sustentación con el Portfolio Manager	Aprobación para la sustentación final sobre los hitos anteriores.	Obligatorio
Hito 5: Sustentación final	Sustentación de todo los entregables de los hitos anteriores.	Obligatorio
Hito 6: Sustentación con el Portfolio Manager	Aprobación para la sustentación parcial sobre el desarrollo del proyecto, el objetivo 3 y la cartera de proyectos.	Obligatorio
Hito 7: Sustentación parcial	Sustentación sobre los hitos anteriores.	Obligatorio
Hito 8: Sustentación con el Portfolio Manager	Aprobación para la sustentación final sobre los hitos anteriores del proyecto.	Obligatorio
Hito 9: Sustentación final	Sustentación final de todos los hitos del proyecto.	Obligatorio

6.6 Plan de gestión de comunicaciones

6.6.1 Matriz de comunicaciones

Tabla 12

Matriz de comunicaciones

Stakeholder	Información	Método	Frecuencia	Remitente
Comité de Proyectos	Grupo de personas encargadas de revisar y aceptar los hitos establecidos en el proyecto.	Exposición parcial y final virtual por Blackboard	Dos veces en el semestre	PMO
Cliente	El PO se encarga de delimitar el alcance y revisar los avances del proyecto, así como de otorgar una retroalimentación de estos.	Reunión virtual por Blackboard	Una vez a la semana	Ángel Velásquez
Portfolio Manager	El PM se encarga de revisar el avance del proyecto y dar	Reunión virtual por Blackboard	Dos veces por semana	Juan Ramírez

	una retroalimentación respecto a los avances.			
Recurso	Recurso Humano de IT Services o Software Factory que apoya en los proyectos de TP1 y TP2	Reunión virtual por Microsoft Teams	Tres veces a la semana	Antony Quispe

6.6.2 Restricciones o Suposiciones de Comunicación

Tabla 13

Restricciones o suposiciones de comunicación

Supuestos	Restricciones
Reuniones con el cliente los martes	Duración 1 hora
Reunión con el Portfolio Manager los martes y sábados	Duración 2 horas los martes y 3 horas los sábados
Exposición con la PMO a la mitad y final del semestre	Duración de 15 minutos por reunión
Reuniones con los recursos los martes, jueves y sábado	Duración 20 minutos

6.6.3 Glosario de Terminología Común

- *PMO: Grupo de personas que son los stakeholders más importantes de todos los proyectos.*
- *PO: Es el rol Product Owner del Framework SCRUM*
- *PM: Es el encargado de guiar en la gestión del proyecto*

7. CONCLUSIONES

1. Se logró validar que las características seleccionadas para la construcción del Game Engine y Framework a partir de los benchmarking de las tablas 4 y 5 del capítulo 4, fueron suficientes para desarrollar un videojuego 2D totalmente funcional, como se ve en la demo realizada como parte de las pruebas de aceptación del capítulo 5.
2. Según el benchmarking de la tabla 4 del capítulo 4, vemos que todos los Game Engines comparados a excepción de Unreal Engine, no cuentan con un Game Framework, lo cual nos indica que la mayoría de estas herramientas no suelen agregarlo como parte de su arquitectura. Esto puede deberse a que incluir un Framework en la arquitectura significa un esfuerzo extra de diseño, implementación, mantenimiento y pruebas como, se evidencia en el capítulo 3 y 4.
3. Como se presentó en la tabla 6 y en el análisis de benchmarking del capítulo 4, el lenguaje de programación más adecuado para el desarrollo de nuestro proyecto es C++, dado que cuenta con las características necesarias para el desarrollo eficiente de videojuegos o herramientas para este sector.
4. Durante el desarrollo del proyecto, logramos construir estructuras complejas para una implementación correcta de los enfoques de arquitectura seleccionadas, tal como se aprecia en el capítulo 4, sección “4.2 Arquitectura del proyecto”. Asimismo, C++ nos ayudó a lograr una integración completa entre los componentes, como se ve en el capítulo 4, sección “4.3 Game Engine y Game Framework”. Por tanto, podemos decir que la selección de C++ según la tabla 6 del capítulo 4 fue adecuada y vital para lograr construir el proyecto exitosamente.
5. El diseño del Game Engine y Framework fue enfocado a la tecnología 2D, lo cual permitió que se redujera la complejidad en las opciones de la interfaz gráfica de usuario, según el 91.6% de encuestados (ver figuras 21 y 22) y en la configuración de las propiedades de los Game Objects, de acuerdo con el 100% de usuarios (revisar figura 23). Esto porque no se agregaron opciones genéricas u orientadas a entornos 3D, los cuales son innecesarias para el proyecto actual. Además, se disminuyó el consumo de recursos, ya que no se cargan y/o consumen capas de abstracción para ambientes 3D, tal como se menciona en las exclusiones del proyecto en el capítulo 1 en la sección “1.6.2 Alcance”.

6. Fue necesario realizar un diseño desde 0, ya que la mayoría de Game Engines no cuentan con la integración de un Game Framework y están enfocados a entornos 3D. Por tanto, el diseño de la arquitectura de los game engines actuales solo nos ayudó como referencia para el análisis de características típicas (ver tabla 4), el funcionamiento en general, patrones de implementación y la interfaz gráfica de usuario.
7. La implementación del Game Engine y Game Framework desde 0, fue una decisión correcta, ya que logramos construir el proyecto según el diseño y atributos de calidad planeados. Dado que al utilizar plantillas o proyectos base hubiera significado una desventaja, porque pueden traer consigo muchos problemas como, posibles bugs, código sin optimizar o desactualizado, malas prácticas, fugas de seguridad, personalización limitada, retos en la integración, funcionalidades innecesarias, entre otros (Dietrich, 2017).
8. Como se presentó en las figuras 21 y 22 la interacción de los encuestas con la interfaz de usuario obtuvo un 91.6% de aceptación. Por tanto, podemos decir que el presente proyecto tuvo un desarrollo exitoso.
9. Como se evidencia en la figura 25, la funcionalidad de física solo obtuvo un 41.6% de aceptación por los usuarios. Esto indica que dicha funcionalidad debe ser mejorada, ya que actualmente es un poco confuso o complicado su configuración.

8. RECOMENDACIONES

En base a la experiencia obtenida durante el desarrollo del proyecto recomendamos lo siguiente:

- Es muy importante el uso de la arquitectura Entity Component System (ECS), ya que esta permite el uso adecuado de la memoria cache L1, L2, L3 de la CPU, lo cual logra un mejor rendimiento del software. Además, el ECS está orientado a los datos y no a objetos, lo cual resulta beneficioso en videojuegos, ya que la programación orientados a objetos según incrementa las funcionalidades del Game Engine, aumenta la complejidad de manejar la jerarquía de objetos y sus relaciones entre sí. Sin embargo, el ECS propone la definición de estructuras sencillas que contienen identificadores, datos y lógica de procesamiento, separados en entidades, componentes y sistemas respectivamente. Esto permite agregar nuevas funcionalidades sin incrementar la complejidad en la lógica de programación del Game Engine y Game Framework.
- El presente proyecto solo se enfoca en el render 2D con OpenGL para Windows, Linux y macOS. Sin embargo, para este último existe Metal, el cual permite un manejo nativo de los recursos gráficos de cualquier dispositivo Apple (Apple, 2023). Por tanto, recomendamos el uso de Metal para cubrir plataformas Apple de forma nativa. Esto se puede lograr gracias al enfoque extensible de nuestro proyecto que permite extender el módulo de gráficos del Game Engine y Game Framework.
- En nuestro proyecto utilizamos C++, por tal razón, se debe evitar la combinación con C puro, ya que esto puede genera en muchas ocasiones comportamiento no previsto. Además, se debe tener presente el uso de adecuado del tiempo de vida de las variables utilizadas.
- Este proyecto ha permitido generar tres propuestas para la cartera de proyectos que se enfocan en ampliar las plataformas que soporta como, WebGL para juegos en browsers, Metal para plataformas Apple y Vulkan para dispositivos Android. Se debe tener en cuenta al momento de desarrollar cualquiera de estos proyectos, las buenas prácticas de C++, el uso de ECS y los patrones de software aplicados a videojuegos con el fin de no disminuir el rendimiento o calidad del proyecto original.

9. [ANEXOS]

Anexo A: WASC

Ensayo 1: Ciudadanía

Resumen

En este documento se sustenta la competencia WASC sobre ÉTICA Y CIUDADANÍA, alineado a nuestro proyecto "Game Engine y Framework Open-Source para el desarrollo de video juegos 2D", ya que todo proyecto tiene un impacto en la sociedad. Además, en nuestro caso, el proyecto permite el desarrollo de juegos debe estar muy claro el tema de ética y las responsabilidades con la sociedad.

Proyecto

Para nuestro proyecto de tesis se ha tomado como tema el realizar una aplicación que permita la construcción de juegos en 2D para desktop.

El proyecto consta de tres secciones:

- Desarrollo de un Game Framework 2D
- Desarrollo de un Game Engine 2D
- Desarrollo de un videojuego 2D utilizando el Game Framework 2D y Game Engine 2D

Se decidió desarrollar un proyecto de este tipo por dos razones:

Por un lado, creemos que va a ser un aporte importante para la carrera de Ingeniería de Software. Esto se debe a que estamos elaborando nuevo conocimiento sobre los beneficios y dificultades al diseñar e implementar un Game Framework y Game Engine 2D. Por otro lado, la problemática que sufren los desarrolladores de juegos 2D independientes con el uso de Game Engines del mercado son diversas como, el costo de la licencia, curva de aprendizaje alta, herramientas enfocadas en juegos 3D con decenas de opciones y la necesidad de recopilar bibliotecas de terceros para satisfacer algunas características 2D. Esto hace que sean poco productivos o que tengan que pagar licencias muy costosas.

Licencia de los softwares

La problemática de nuestro proyecto es la siguiente: "La oferta de game engines para videojuegos 2D open source es limitada, por lo cual los desarrolladores tienden a utilizar e integrar por cuenta propia diversas configuraciones, herramientas y bibliotecas de distintas fuentes para construir este tipo de videojuegos.". Esto genera algunas implicaciones en el ambiente ético como ciudadano a nivel mundial, las cuales se mencionan a continuación.

Al tener la necesidad de integrar o utilizar distintas herramientas o bibliotecas de terceros los desarrolladores pueden cometer el error de mezclar licencias o seleccionar aquellas que no son permisivas en el sector comercial. Esto se debe a que un software open-source no garantiza que puede ser utilizado o modificado para el uso comercial sin un retorno. Por ejemplo, GPL indica explícitamente que todo trabajo que utilice bibliotecas o herramientas con esta licencia debe de distribuir el código fuente por completo bajo la misma licencia (MDN Web Docs, 2021), lo cual hace difícil su comercialización. Asimismo, las normas de la GNU para licencias GPL y LGPL son aplicables en cualquier parte del mundo para todos aquellos que son desarrolladores, arquitectos o analistas de software. Además, son conductas establecidos por más de tres décadas en el área del software que deben ser respetadas por todos los que lo integran.

Por lo presentado, hemos identificado los siguientes actores en la problemática del proyecto.

Por un lado, los desarrolladores del Game Engine y Game Framework, los cuales tienen la obligación de redactar correctamente todas las licencias de herramientas y bibliotecas utilizadas para la construcción del proyecto. Dicha redacción debe expresar claramente la licencia que rige el proyecto, así como las pertenecientes a componentes de terceros. Asimismo, tiene que mostrar en qué casos está permitido o no usarlo, tales como, uso comercial, no comercial, tipo de juegos, etc. Por otro lado, los autores de juegos tienen la obligación de leer con cuidado y de ser posible consultar con un abogado sobre la restricciones y libertades de la licencia con la cual son lanzados el Game Engine y Game Framework. Esto con el fin de no infringir los derechos de autor y la licencia en sí.

Desarrollo de Videojuegos inapropiados

Desde que inició el desarrollo de videojuegos y su comercialización se han construido títulos de juegos con distintas temáticas como, el terror, horror, fantasía, comedía, acción, violencia extrema, musical, ciencia ficción, entre otros. Sin embargo, han existido y aún existen algunos estudios desarrolladoras de videojuegos, especializados en temáticas muy desagradables como, la violación, canibalismo, asesinato en serie, simuladores de acoso sexual, entre otros temas relacionados. Por tal razón, nosotros como autores de un Game Engine y Game Framework debemos tratar de disminuir este tipo de títulos, para ello proponemos colocar cláusulas de uso de nuestro proyecto. Es decir, dentro de nuestra licencia se prohíbe la realización de juegos con argumentos iguales o parecidas a las descritas anteriormente.

Pluralismo

Para la realización de este proyecto se tuvo que realizar una exhaustiva investigación, así como el intercambio de ideas entre diversos perfiles. Por un lado, las reunión e intercambio de opiniones entre el PO y PM con nosotros, ya que ellos tienen distintos perfiles en cuanto carreras. Asimismo, su amplia experiencia nos ha ayudado a plantear de la mejor manera nuestro proyecto. Desde la investigación hasta implementación y validación del proyecto. Por otro lado, a través de la web concretamente foros de discusiones de desarrolladores de videojuegos y artículos científicos logramos recoger diversas opiniones sobre cómo desarrollar videojuegos, los pros y contra de usar una tecnología u otra. Asimismo, tomamos recomendaciones e interpretaciones sobre el uso de licencias de software desde el punto de vista legal para que nuestro proyecto no tenga ninguna deficiencia en ese sentido. También, hemos podido visualizar opiniones de sobre el uso indebido de la tecnología que permite el desarrollo de videojuegos.

Importancia del diálogo

Nuestro proyecto posee una alta complejidad por el lado de la arquitectura lógica, así como en los patrones de software a emplear. Por tal razón, se realizó una búsqueda amplia sobre enfoques más adecuados para la construcción del proyecto. Tales como, arquitectura Entity Component System (ECS), modular, middleware, singleton, doble colisión, entre otros.

En este sentido, cada uno de los enfoques a utilizar fueron investigados para su sustentación y discutidos con nuestro PO, PM y entre los miembros del equipo. Esto con el fin de alcanzar un consenso de las tecnologías, patrones y enfoques arquitectónicos a

utilizar. Por tal razón, creemos que el dialogo y el respeto mutuo es muy importante, ya que nos permite intercambiar ideas y mejorar el proyecto en sí.

Dado a que nuestro proyecto utiliza bibliotecas para realizar algunas funcionalidades, como la física, el render, entre otros. Fue vital discutir el tema de las licencias tanto entre partners como con nuestro cliente con el objetivo de no mezclar licencias o agregar por equivocación los del tipo GLP o LGPL.

Derechos y deberes

Deberes de los autores del proyecto:

- Respetar completamente los acuerdos de las licencias de cada biblioteca o componente de terceros que conforman el proyecto
- Redactar claramente sin ambigüedades las restricciones y libertades de la licencia del proyecto. Asimismo, mencionar explícitamente si tiene o no un uso comercial, así como si se basa en alguna licencia en particular y si ha sido modificada.
- Prohibir el desarrollo de videojuegos inmorales a través de nuestra tecnología.

Derechos de los autores del proyecto:

- Ser reconocidos como los autores del Game Engine y Game Framework en su totalidad excepto por las herramientas de terceros utilizados.
- Los usuarios que utilicen el proyecto deben de respetar la licencia proporcionada al proyecto y reconocer o dar los créditos a los autores si se especifica en la licencia como tal.
- El usuario que utilicen el proyecto debe de respetar las cláusulas sobre la prohibición de creación de videojuegos inmorales.

Deberes de los usuarios del proyecto:

- Leer atentamente las obligaciones y libertades que se describe en la licencia del proyecto con el fin de respetarlas.
- Cumplir por completo con las indicaciones de la licencia.

Derechos de los usuarios del proyecto:

- Contar con una copia original del proyecto, que esta sea clara y sencilla de comprender sin ambigüedades.

- Ser notificado cuando se realice un cambio en los términos en la licencia y que estos apliquen desde 1 mes después de su publicación en adelante sin afectar al producto o productos que fueron construidos con la licencia anterior.

Conclusiones

En la industria de Software es muy importante tener claro las diferencias de las licencias open-source, ya que cada una de ellas tienen sus propias libertades y prohibiciones, y que si se infringen puede ser causal de demanda por el dueño del software.

En la industria de los videojuegos existen muchos estudios como personas independientes que crean cualquier tipo de videojuegos que pueden ir desde los más inofensivo a lo inmoral. Por ello, los desarrolladores de tecnología para la creación de videojuegos deben tener en cuenta este tipo de personas y tratar de evitarlos en la mayor manera posible.

En un proyecto tan complejo con este es de vital importancia el intercambio de ideas entre personas de distintos perfiles para encontrar un balance y dirigir correctamente el enfoque del trabajo, así como su gestión.

Cualquier tipo de proyecto debe siempre mostrar de forma transparente las obligaciones y libertades de su licencia con el fin de no vulnerar los deberes y derechos tanto de los desarrolladores como los usuarios.

Ensayo 2: Ciudadanía

El tema propuesto en este trabajo de investigación fue el diseño e implementación de una aplicación para desktop y/o laptop del tipo Game Engine y Game Framework para el desarrollo de juegos en 2D de forma más fácil y eficiente. Por ello, en el caso de ética y ciudadanía que trata nuestro proyecto viene a ser la libertad de expresión que ofrecemos con nuestra herramienta. En el sentido de que nuestra herramienta permite la creación de juegos de cualquier tipo mientras sea 2D, esta libertad puede resultar riesgoso para los usuarios que puedan crear juegos que atenten contra lo moral o que traten temas que violan la ley, o también apropiarse de derechos de autor de recursos que no son suyos como sprites o tile maps que no sean de libre acceso.

Empezando con la definición de ciudadanía según Marshall (1997), lo determina como la condición de un ser humano de poseer derechos civiles, económicos, culturales, políticos y sociales, además de los deberes que debe cumplir en la sociedad. Asimismo, el problema identificado que atenta contra la ética y ciudadanía vendría a ser el uso inadecuado de nuestra propuesta, ya que se pueden desarrollar videojuegos 2D que atenten contra la ley de protección de derechos y salud mental o que traten temas sensibles para la sociedad, como la violación, el aborto, entre otros. También, el uso de licencias no permisivas o de uso no comercial. Por tanto, para evitar todos los problemas mencionados anteriormente se definió modificar la licencia open-source con una prohibición de desarrollo de videojuegos con temáticas que atenten contra los derechos de los ciudadanos, además de ser explícitos en listar las licencias de herramientas de terceros utilizados en la implementación del presente proyecto.

En cuestión de Pluralismo, durante el desarrollo de nuestro proyecto hubo un constante intercambio de opiniones y dialogo entre los dos integrantes del grupo y nuestros asesores. Siempre nos reunimos para acordar las tecnologías y realizar las comparaciones entre estas, además de tener las tareas asignadas para cada uno y revisar constantemente para evitar retrasos. También, en cuestiones de solidaridad, siempre que uno de los integrantes tenía problemas para avanzar alguna parte del diseño, desarrollo o documentación del proyecto, podía pedir apoyo a su partner.

Finalmente, el concepto de ciudadanía ayuda a tener un valor agregado al proyecto sobre la solución de algún conflicto ético como el explicado en este ensayo. Y también teniendo en cuenta la ética y el constante dialogo nos permitió desarrollar este proyecto de la mejor manera y sin problemas.

Anexo B: Atributos de calidad

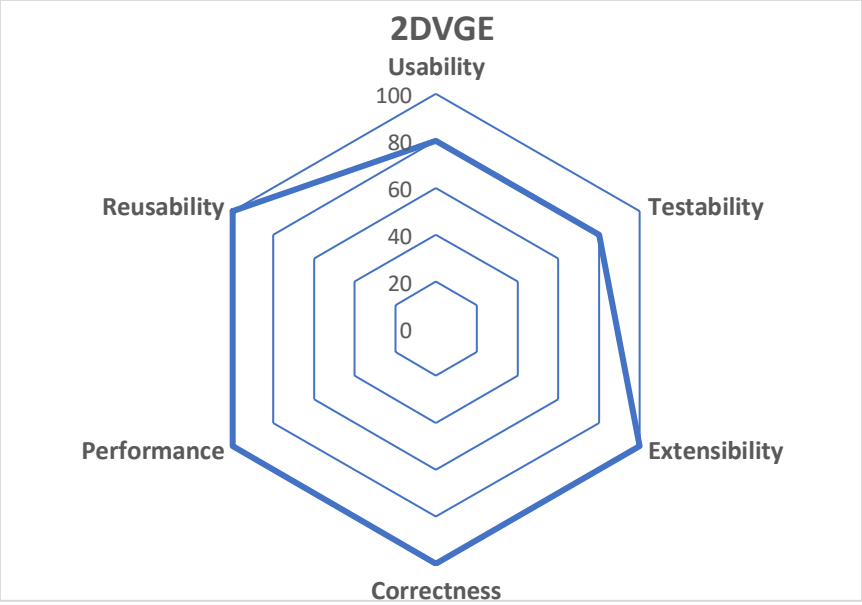
Los atributos de calidad son características que poseen un software en términos de requisitos no funcionales (QuBar, 2012). Estos forman parte importante junto a los requisitos funcionales para la definición de los drivers de arquitectura de software (Mpiua, 2014).

A continuación, se evidencia el proceso realizado para la selección de los atributos de calidad para el proyecto "Game Engine y Framework open-source para el desarrollo de videojuegos 2D". Asimismo, la definición de cada uno de ellos y finalmente las tácticas para implementarlas.

Taxonomía

Figura 39

Taxonomía



Nota. Basado en "CISC 877 Video Game Development Week 2: Software Architecture of Computer Games", por Graham, 2010 (https://research.cs.queensu.ca/home/graham/cisc864/notes/swarch.pdf) y en "Toward Quality-Driven Development of 3D Computer Games", por Graham & Roberts, 2007 (https://doi.org/10.1007/978-3-540-69554-7_20).

Atributos de calidad seleccionados

En este apartado se muestran los atributos que se han seleccionado como los más relevantes para el desarrollo de videojuegos, Game Engines y Game Frameworks.

La selección de los atributos de calidad fue realizada en base a las recomendaciones de Nicholas Graham en su clase de desarrollo de videojuegos y arquitectura (Graham, 2010).

- **Performance:** Consumo de recursos (CPU, Memoria, etc) del software. Además, el tiempo que tarda en procesar las tareas solicitadas.
- **Correctness:** Implementación de requisitos funcionales exitosamente. También, la identificación de bugs en el software de forma sencilla mediante un conteo de bugs/1000.
- **Usability:** Facilidad de usar el software desde el punto de vista del usuario final. Esto implica un GUI muy sencillo y un manual de uso de pocas páginas.
- **Availability:** Capacidad del software de estar disponible para ser usado por los usuarios en cualquier momento.
- **Testability:** Facilidad del software para ser probado, lo cual es muy importante, ya que esta tarea ayuda a encontrar bugs y a aumentar la exactitud (Correctness). Asimismo, se refiere a la cantidad de pruebas ejecutas y el tiempo que tardan.
- **Extensibility**. Un software con esta característica permite agregar nuevas funcionalidades sin comprometer sus demás componentes o partes del sistema. Esta muy ligado al bajo acoplamiento, ya que, si este es muy alto, una extensión de la característica de un componente afecta a otros partes lo cual dificultaría la tarea de agregar nuevas funcionalidades.
- **Reusability**: Software construido a partir de partes o componentes que puede ser utilizados en distintas secciones, lo cual ayuda a disminuir el código repetido.

Tácticas aplicadas por atributo de calidad

En este apartado se muestran las tácticas o estrategias a realizar para cumplir con cada atributo de calidad. Estas estrategias son presentadas en base a las recomendaciones de T.C. Nicholas Graham y Will Roberts en su artículo científico (Graham & Roberts, 2007).

Tabla 14

Tácticas

Código	Tácticas
01	Interfaz de usuario intuitivo para el desarrollo de juegos

02	Software con arquitectura modular
03	Lenguaje de programación C++ y scripts de bajo nivel
04	Uso de técnicas multi-hilo
05	Software con arquitectura ECS y Scene Management
06	Definición clara de las Historias de Usuario
07	Pruebas unitarias
08	Manejo de errores o excepciones
09	Software con arquitectura Middleware
10	Software construido a partir de un Framework

Nota. Basado en "Toward Quality-Driven Development of 3D Computer Games", por Graham & Roberts, 2007 (https://doi.org/10.1007/978-3-540-69554-7_20).

Tabla 15

Tácticas por atributo de calidad

Atributos de calidad	Tácticas
Performance	03, 04, 05
Correctness	06, 07
Usability	01, 08
Testability	02, 05, 07
Extensibility	02, 05, 09, 10
Reusability	01, 02, 05, 10

Nota. Basado en "CISC 877 Video Game Development Week 2: Software Architecture of Computer Games.", por Graham, 2010 (https://doi.org/10.1007/978-3-540-69554-7_20).

Anexo C: Costos y presupuestos

1. Introducción

1.1 Propósito

EcoSys (2022) señala que el plan de gestión de costos es un proceso esencial antes de iniciar un proyecto, ya que nos ayuda a realizar un cálculo y presupuesto preliminar sobre lo que costará construir el proyecto. Asimismo, para que el proyecto finalice con éxito debe de haber cumplido con ciertos criterios como, la implementación total de los requisitos funcionales y no funcionales, cumplimiento de las actividades dentro del cronograma, costos dentro del cálculo o presupuesto definido.

1.2 Alcance

Recursos en Project Management (2014) indica que los costes internos son aquellos que están relacionados con la realización de tareas con recursos de la organización. Asimismo, los costes externos son aquellos generados por actividades hechas mediante colaboradores externos. Por tanto, el presente proyecto contempla costos internos y externos dentro de su alcance.

A continuación, presentamos los costes agrupados según su tipo:

Interno

- Recursos humanos del Proyecto y del equipo
- Contratando equipo adicional
- Costo capital del equipo a usar
- Programas de Software y licencias

Externo

- Costos de Infraestructura

Tabla 16

Costos de infraestructura

Duración del proyecto:	8 meses					Tipo de Cambio	PEN 4,00
Costo de Hardware, Software y Personal	**Cantidad**	**Precio unitario en dólares**	**Precio total en soles por cantidad**	**Horas**	**Meses**	**Costo total en soles por cantidad y tiempo**	
Recursos de Hardware							
Laptop (AMD RYZEN 5 , 16 GB RAM, NVIDIA GeForce GTX 1650)	1	$ 1.250,00	PEN 4.750,00	-	-	PEN	4.750,00
PC (Intel Core i7, 16 GB RAM, Radeon 5600 XT)	1	$ 1.200,00	PEN 4.560,00	-	-	PEN	4.560,00
Costo total de Recursos de Hardware		**$ 2.450,00**	**PEN 9.310,00**			**PEN**	**9.310,00**
Recursos de Software							
Github PRO	1	$ 10,00	PEN 40,00	-	8	PEN	320,00
Jira	1	$ 10,00	PEN 40,00	-	8	PEN	320,00
Licencias Windows	2	$ 15,00	PEN 60,00	-	-	PEN	120,00
Clion	2	$ 19,90	PEN 79,60	-	8	PEN	159,20
2D Beat'm up Assets	1	$ 15,00	PEN 60,00	-	-	PEN	60,00
Costo total de Recursos de Software		**$ 15,00**	**PEN 60,00**			**PEN**	**979,20**
Recursos de Infraestructura							
Servicios Energia Electrica e Internet	1	$ 45,00	PEN 180,00	-	8		1.440,00
Costo total de Infraestructura		**$ 45,00**	**PEN 180,00**			**PEN**	**1.440,00**
Recursos Humanos							
Project Manager	1	$ 10,00	PEN 38,00	160	-		6.080,00 PEN
Scrum Master	1	$ 10,00	PEN 38,00	160	-		6.080,00 PEN
Software Developer Junior	2	$ 7,00	PEN 26,60	560	-		14.896,00 PEN
QA Analyst	1	$ 7,00	PEN 26,60	47	-		1.250,20 PEN
Research Analyst	1	$ 7,00	PEN 26,60	86	-		2.287,60 PEN
Portfolio Manager	1	$ 7,00	PEN 28,00	86	-		2.408,00 PEN
Product Owner	1	$ 7,00	PEN 28,00	86	-		2.408,00 PEN
Costo total de Recursos Humanos		**$ 55,00**	**PEN 211,80**			**PEN**	**35.409,80**
Costo total		**$ 2.550,00**	**PEN 9.701,80**			**PEN**	**47.139,00**

2. Costos Internos y Externos

El costo total calculado a lo largo de los 8 meses del proyecto fue 47.139,00 soles, con costo de adquisición de hardware de 9.310,00 soles, software con un total de 979,20 soles, recursos de infraestructura con un total de 1.440,00 soles y los recursos humanos o personas involucradas en el desarrollo del proyecto se previó un total de 35.409,80 soles. Luego se realizó un análisis de ingresos que producirían nuestro proyecto mensualmente, junto a los gastos mensuales que se predicen para mantener el proyecto.

Tabla 17

Gastos

Gastos	Precio unitario en dólares	Precio total en soles por cantidad	Tipo	Cantidad* Mes	Gastos Mensual
Soporte					
Personal	$ 500,00	PEN 2.000,00	Mensual	-	PEN 2.000,00
Gatos Totales por Soporte	**$ 500,00**	**PEN 2.000,00**			**PEN 2.000,00**
Gastos Totales					**PEN 2.000,00**

Nota. Los gastos previstos cubren lo que es el soporte de las licencias pagadas que se ofrecerán con el software.

Tabla 18

Métodos de ingreso

Métodos de Ingresos	Precio unitario en dólares	Precio total en soles por cantidad	Tipo	Cantidad* Mes	Ingreso Mensual
Licencias					
Comunnity	$ -	PEN -	Open Source	-	
Professional	$ 20,00	PEN 80,00	Licencia Mensual	PEN 10,00	PEN 800,00
Enterprise	$ 30,00	PEN 120,00	Licencia Mensual	PEN 10,00	PEN 1.200,00
Ingresos Totales por Licencias	**$ 50,00**	**PEN 200,00**			**PEN 2.000,00**
Cursos					
Beginner Game Developer whit 2DMOVOGEngine	$ 20,00	PEN 80,00	Pago Único	PEN 15,00	PEN 1.200,00
Intermediate Game Developer whit 2DMOVOGEngine	$ 20,00	PEN 80,00	Pago Único	PEN 15,00	PEN 1.200,00
Advanced Game Developer whit 2DMOVOGEngine	$ 20,00	PEN 80,00	Pago Único	PEN 15,00	PEN 1.200,00
Ingresos Totales por Cursos	**$ 20,00**	**PEN 80,00**			**PEN 7.600,00**
Ingresos Totales					**PEN7.600,00**

Nota. Los ingresos definidos vienen a ser por la venta de licencias del software y cursos para especialización en el Game Engine.

3. Enfoque de gestión de costes

3.1 Indicadores de Costes

Tabla 19

Indicadores de coste

Mes	Ingreso	Gasto/Inversion	Ganancia	ROI	Ganancia Acumulada
0	PEN -	PEN 47.139,00	$ -47.139,00	-1,00	$ -
1	PEN 7.600,00	PEN 2.000,00	$ -41.539,00	-0,88	$ 5.600,00
2	PEN 7.600,00	PEN 2.000,00	$ -35.939,00	-0,76	$ 11.200,00
3	PEN 7.600,00	PEN 2.000,00	$ -30.339,00	-0,64	$ 16.800,00
4	PEN 7.600,00	PEN 2.000,00	$ -24.739,00	-0,52	$ 22.400,00
5	PEN 7.600,00	PEN 2.000,00	$ -19.139,00	-0,41	$ 28.000,00
6	PEN 7.600,00	PEN 2.000,00	$ -13.539,00	-0,29	$ 33.600,00
7	PEN 7.600,00	PEN 2.000,00	$ -7.939,00	-0,17	$ 39.200,00
8	PEN 7.600,00	PEN 2.000,00	$ -2.339,00	-0,05	$ 44.800,00
9	PEN 7.600,00	PEN 2.000,00	$ 3.261,00	0,07	$ 50.400,00
10	PEN 7.600,00	PEN 2.000,00	$ 8.861,00	0,19	$ 56.000,00
11	PEN 7.600,00	PEN 2.000,00	$ 14.461,00	0,31	$ 61.600,00
12	PEN 7.600,00	PEN 2.000,00	$ 20.061,00	0,43	$ 67.200,00
13	PEN 7.600,00	PEN 2.000,00	$ 25.661,00	0,54	$ 72.800,00
14	PEN 7.600,00	PEN 2.000,00	$ 31.261,00	0,66	$ 78.400,00
15	PEN 7.600,00	PEN 2.000,00	$ 36.861,00	0,78	$ 84.000,00

Figura 40

Ganancia

Nota. Realizando el análisis de la evolución de las ganancias a lo largo de los meses se identificó que para el mes 10, se obtendrían ganancias sobre lo invertido inicialmente.

Figura 41

Retorno de inversión

ROI (ROI = Ganancia – Inversión/Inversión)

Nota. El ROI obtenido conforme a la fórmula a través de los meses coincide con el gráfico de las ganancias que demuestran que en el mes 10 se obtendría un ROI de 0.07, definiendo así la recuperación de lo invertido.

10. REFERENCIAS BIBLIOGRÁFICAS

ACM Ethics. (2016). *Ingeniería de Software Código de Ética y Práctica Profesional 5.2*. Recuperado el 30 de junio de 2023, de https://ethics.acm.org/wp-content/uploads/2016/07/SE-code-spn.pdf

Ali, Z., & Usman, M. (2016). A framework for game engine selection for gamification and serious games. *2016 Future Technologies Conference (FTC)*, 1199-1207. https://doi.org/10.1109/ftc.2016.7821753

Andrade, A. (2015). Game engines: a survey. *EAI Endorsed Transactions on Serious Games, 2*(6). https://doi.org/10.4108/eai.5-11-2015.150615

Apple. (2023). *Accelerate graphics and much more with Metal*. Apple Developer. Recuperado el 30 de junio de 2023, de https://developer.apple.com/metal/

Box2D. (2015). *A 2D physics engine for games*. Recuperado el 01 de abril de 2021, de https://box2d.org/documentation/

Bryant, B., & Saiedian, H. (2021). A State Saturation Attack against Massively Multiplayer Online Videogames. *In Proceedings of the 7th International Conference on Information Systems Security and Privacy*, *1*, 217-225. https://doi.org/10.5220/0010302002170225

Castrejón, R., & Aliaga, D. (2020). *Social Network Marketing con relación al Customer Engagement y Brand Loyalty de las empresas del sector gaming, en jóvenes de Lima* [Trabajo de investigación, Universidad Peruana de Ciencias Aplicadas]. Repositorio Académico UPC. http://hdl.handle.net/10757/654543

Congreso de la República del Perú. (2011). *Ley 29733 de 2011. Por lo cual se expide Ley de protección de datos personales.*

Consulting Tutisani. (2018). *Modular Software Architecture*. Recuperado el 01 de abril de 2021, de https://www.tutisani.com/software-architecture/modular-software-architecture.html

Cortizo, J. (2018). *Si hay una industria que no es un juego, esa es la de los Videojuegos*. Product Hackers. Recuperado el 01 de abril de 2021, de https://en.digital/blog/videojuegos-industria-mobile-crecimiento

Dietrich, E. (2017). *Code Reuse is Not a Good Goal*. NDepend. Recuperado el 29 de junio de 2023, de https://blog.ndepend.com/code-reuse-not-good-goal/

EcoSys. (2022). *Gestión de costos del proyecto: Pasos, conceptos básicos y beneficios*. Hexagon. Recuperado el 09 de julio de 2023, de https://www.ecosys.net/es/conocimientos/gestion-de-costos-del-proyecto-pasos-conceptos-basicos-y-beneficios/

Espinal, S., Miramira, B., & Velásquez, Á. (2022). Open-Source Game Engine & Framework for 2D Game Development. *2022 IEEE Engineering International*

Research Conference (EIRCON), 1-4. https://doi.org/10.1109/EIRCON56026.2022.9934816

Galvan, A. (2018). *Game Engine Architecture.* Alain.xyz. Recuperado el 18 de setiembre de 2021, de https://alain.xyz/blog/game-engine-architecture

Gascón, O. (2017). *Determinar el presupuesto*. TodoPMP. Recuperado el 09 de julio de 2023, de https://todopmp.com/determinar-el-presupuesto/

GeeksForGeeks. (2019). *What's the difference between Scripting and Programming Languages?*. Recuperado el 01 de abril del 2021, de https://www.geeksforgeeks.org/whats-the-difference-between-scripting-and-programming-languages/

G&M News. (2020). *Perú: Un mercado prometedor para los videojuegos y los Esports*. Recuperado el 01 de abril de 2021, de https://g-mnews.com/news/peru-un-mercado-prometedor-para-los-videojuegos-y-los-esports/

Gibson, J. (2022). *Introduction to Game Design, Prototyping, and Development* (3.ª ed.). Addison-Wesley Professional. https://www.informit.com/store/introduction-to-game-design-prototyping-and-development-9780136619963

GLFW. (2022). *Introduction*. Recuperado el 13 de julio de 2023, de https://www.glfw.org/docs/3.3/index.html

GLM. (2020). *OpenGL Mathematics*. Recuperado el 20 de marzo de 2021, de https://glm.g-truc.net/0.9.9/

Graham, N., & Roberts, W. (2007). Toward quality-driven development of 3D computer games. *Lecture Notes in Computer Science, 4323*, 248–261. https://doi.org/10.1007/978-3-540-69554-7_20

Graham, N. (2010). *CISC 877 Video Game Development Week 2: Software Architecture of Computer Games*. Queen's University. Recuperado el 24 de marzo de 2021, de https://research.cs.queensu.ca/home/graham/cisc864/notes/swarch.pdf

Gregory, J. (2018). *Game Engine Architecture* (3.ª ed.). A K Peters/CRC Press. https://www.routledge.com/Game-Engine-Architecture-Third-Edition/Gregory/p/book/9781138035454

Interxion. (2019). *Los videojuegos revolucionan el tráfico de internet*. Recuperado el 01 de abril de 2021, de https://www.interxion.com/es/blogs/2019/09/los-videojuegos-revolucionan-el-trafico-de-internet

Joshi, A., Kale, S., Chandel, S., & Pal, D. (2015). Likert Scale: Explored and Explained. *Current Journal of Applied Science and Technology, 7*(4), 396–403. https://doi.org/10.9734/BJAST/2015/14975

Juárez, E. (2016). *La gestión del Alcance del proyecto*. AprenderCompartiendo. Recuperado el 09 de julio de 2023, de https://aprendercompartiendo.com/gestion-del-alcance-del-proyecto/

Khronos Group. (2017). *The Industry's Foundation for High Performance Graphics*. Recuperado el 20 de marzo de 2021, de https://www.khronos.org/opengl/

Khronos Group. (2021). *OpenGL Loading Library*. Recuperado el 20 de marzo de 2021, de https://www.khronos.org/opengl/wiki/OpenGL_Loading_Library#glLoadGen

La guía PMBOK. (2013). *Gestión de los costes del proyecto*. uacm123. Recuperado el 09 de julio de 2023, de https://uacm123.weebly.com/3-gestioacuten-de-los-costes-del-proyecto.html

Lin, W. (2020). *Entity Component System for Unity: Getting Started.* Kodeco. Recuperado el 18 de setiembre de 2021, de https://www.kodeco.com/7630142-entity-component-system-for-unity-getting-started

Marshall, T. (1997). Ciudadanía y clase social. *REIS: Revista Española de Investigaciones Sociológicas*, 297-346. http://www.elsolardelasartes.com.ar/pdf/702.pdf

Masood, M., & Wijdan, A. (2020). *Comparison of Programming Languages in Game Development.* ResearchGate. Recuperado el 25 de junio de 2021, de https://www.researchgate.net/publication/342804594_Comparison_of_Programming_Languages_in_Game_Development

MDN Web Docs. (2021). *GPL*. Recuperado el 25 de junio de 2021, de https://developer.mozilla.org/es/docs/Glossary/GPL

Mpiua. (2014). *Atributo de la calidad del software*. Recuperado el 24 de marzo de 2021, de https://mpiua.invid.udl.cat/usabilidad/atributo-de-la-calidad-del-software/

Northglen News. (2020). *The Rise of Massive Multiplayer Online Games, eSports, and Game Live Streaming.* Recuperado el 01 de abril de 2021, de https://northglennews.co.za/249179/the-rise-of-massive-multiplayer-online-games-esports-and-game-live-streaming/

Ocornut, O. (2014). *Dear ImGui*. GitHub. Recuperado el 23 de octubre de 2021, de https://github.com/ocornut/imgui

O'Shea, Z., & Freeman, J. (2019). Game design frameworks: where do we start?. *In Proceedings of the 14th International Conference on the Foundations of Digital Games (FDG '19)*, 1-10. https://doi.org/10.1145/3337722.3337753

Pavkov, S., Franković, I., & Hoić-Božić, N. (2017). Comparison of game engines for serious games. *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 728–733. https://doi.org/10.23919/MIPRO.2017.7973518

Petridis, P., Dunwell, I., Panzoli, D., Arnab, S., Protopsaltis, A., Hendrix, M., & Freitas, S. (2012). Game Engines Selection Framework for High-Fidelity Serious Applications. *International Journal of Interactive Worlds*, *2012*, 1–19. https://doi.or

g/10.5171/2012.418638

QuBar. (2012). *Atributos de calidad*. Recuperado el 24 de marzo de 2021, de http://wiki.uqbar.org/wiki/articles/atributos-de-calidad.html

Raffaillac, T., & Huot, S. (2019). Polyphony: Programming Interfaces and Interactions with the Entity-Component-System Model. *Proceedings of the ACM on Human-Computer Interaction, 3*(8), 1–22. https://doi.org/10.1145/3331150

Recursos en Project Management. (2014). *Tipos de costes en proyectos*. Recuperado el 09 de julio de 2023, de https://www.recursosenprojectmanagement.com/tipos-de-costes/

Statista. (2017). *Number of indie games released on Steam worldwide from 2015 to 2017*. Recuperado el 01 de abril de 2021, de https://www.statista.com/statistics/809258/unmber-indie-games-steam/

SWIG. (2019). *Scripting Languages*. Recuperado el 01 de abril de 2021, de http://www.swig.org/Doc3.0/Scripting.html

Thorn, A. (2011). *Game engine design and implementation* (1ª ed.). Jones & Bartlett Learning. https://library.villanova.edu/Find/Record/1279339

University of Waterloo. (2019). *Plugin architecture*. Recuperado el 01 de abril de 2021, de https://cs.uwaterloo.ca/~m2nagapp/courses/CS446/1195/Arch_Design_Activity/PlugIn.pdf

VG Insights. (2020a). *Are indie games priced too low?*. Recuperado el 18 de marzo de 2021, de https://vginsights.com/insights/article/should-developers-charge-more-for-an-indie-game

VG Insights. (2020b). *Infographic: Indie game revenues on Steam*. Recuperado el 13 de abril de 2021, de https://vginsights.com/insights/article/infographic-indie-game-revenues-on-steam

Video Game Technologies. (2022). *Scene Management and Rendering*. Recuperado el 04 de octubre de 2022, de https://www.di.ubi.pt/~agomes/tjv/teoricas/06-scenemanagement.pdf

Villavicencio, W. (2020). *Gestión del Cronograma según el PMBOK*. Recuperado el 09 de julio de 2023, de https://waltervillavicencio.com/gestion-del-cronograma-segun-el-pmbok/

Wiebusch, D., & Latoschik, M. (2015). Decoupling the entity-component-system pattern using semantic traits for reusable realtime interactive systems. *IEEE 8th Workshop on Software Engineering and Architectures for Realtime Interactive Systems (SEARIS)*, 25-32. https://doi.org/10.1109/SEARIS.2015.7854098

Wozniewicz, B. (2019). *The Difference Between a Framework and a Library*. FreeCodeCamp. Recuperado el 15 de setiembre de 2021, de https://www.freecodecamp.org/news/the-difference-between-a-framework-and-a-library-bd133054023f/