

Facultad de Matemáticas
Programación Estructurada
Proyecto Final. Avances del Proyecto
Docente: Emilio Gabriel Rejón Herrera



Propuesta de proyecto
Videojuego Sobre Batallas Estratégicas Entre Dos Jugadores Que Utiliza Mecánicas
De Lanzamiento De Projectiles Mediante Cañones Para Eliminar Las Tropas Del
Oponente (“Abbys Battle”)

C-Force

Facultad de Matemáticas UADY

Grupo D

Martes 30 de abril del 2024

Integrantes de C-Force:

Alonzo Palacios Rodrigo

Cuevas García Braulio Samuel

Martínez Martínez José Pablo

Moo Pan Jareth Jaziel

Índice

<i>Índice</i>	2
<i>1.0 Antecedentes de la Propuesta</i>	4
<i>2.0 Descripción del Producto de Software</i>	7
<i>3.0 Objetivo General</i>	9
<i>4.0 Objetivos Específicos del Sistema</i>	9
<i>5.0 Diagrama de Casos de Uso</i>	10
<i>6.0 Descripción de Tipos de Usuarios</i>	11
6.1 Usuario Principal (Jugador)	11
6.1.1 Descripción	11
6.1.2 Responsabilidades	11
6.1.3 Características	11
6.1.4 Objetivos	11
6.2 Administrador	11
6.2.1 Descripción	11
6.2.2 Responsabilidades	11
6.2.3 Características	12
6.2.4 Objetivos	12
<i>7.0 Interfaces de Usuario</i>	13
7.1 Interfaces de menú	13
<i>8.0 Definición del estándar de codificación</i>	17
<i>9.0 Descripción de requerimientos funcionales y no funcionales</i>	20
<i>10.0 Descripción de los casos de uso</i>	25
Caso 1: Acceso al menú principal	25
Caso 2: Ingresar al juego	26
Caso 3: Selección de aspecto	28
Caso 4: Tutorial	29
Caso 5: Pausar la partida	31
Caso 6: Turnos de juego	32
<i>11.0 Matriz de Requerimientos</i>	34
<i>12.0 Proceso de Desarrollo</i>	40
12.1 Herramientas utilizadas	40
12.1.1 Dev C++	40

12.1.3 Figma	40
12.1.4 GitHub – GitKraken	40
12.2 Organización	40
12.2.1 WhatsApp – Discord	40
12.2.2 To do List	40
12.3 Monitoreo	41
12.3.1 GitHub – GitKraken	41
12.4 Bitácoras	41
12.5 Medición del trabajo grupal	41
12.6 Medición del trabajo individual	42
13.0 Avances del Código	43
14.0 Reporte de evaluación basada en objetivos	48
Referencias	50

1.0 Antecedentes de la Propuesta

Al realizar este proyecto de software se consideró que en este mundo ya se han explorado un sin número de ideas, por lo que se puso como objetivo tomar una idea y reinventarla para funcionar en el lenguaje de programación C. En base a lo anterior se realizó una lluvia de ideas para encontrar que tipo de proyecto se realizaría, al final se optó por la realización de un videojuego como el producto de software.

Al escoger el tema que se planeaba realizar, surgió una sencilla, aunque no menos interesante problemática que se puso sobre la mesa. Es común que cuando un grupo de dos o más amigos no cuentan con acceso a internet en donde se ubican, sea su casa o algún lugar donde se encuentren de invitados, entre otros, suelen entrar a pie los juegos de mesa para matar el tiempo, pero si no hay suficiente espacio para jugar de forma física, o incluso si no cuentan con ningún juego de mesa físico, una alternativa serían juegos sencillos como “Gato” o “Conecta 4” que se pueden jugar con papel o un lápiz, sin embargo, actualmente muchas personas ya cuentan con dispositivos electrónicos como pueden ser celulares, por ello este proyecto pretende tomar la problemática anterior y proporcionar un videojuego que sea divertido para los usuarios.

El reto de realizar este proyecto de software sería realizarlo en lenguaje C, ya que es estructurado y no orientado a objetos como lo es C++. El producto de software será un juego de video para dos personas, que consiste en un enfrentamiento 1v1 en un tablero designado por casillas, con bases en el popular juego “Battle ship”. Para este se decidió utilizar una temática “Medieval” para el diseño de “Personajes” y “Ambiente”.

Por último, ya que el producto está basado en el videojuego “Battle Ship” es evidente que este resulta nuestra principal competencia, por ello es importante resaltar cuáles son las características de dicho software.

En internet existen un sin fin de versiones del juego de video “Battle ship” alojadas en páginas web como un juego gratuito. A pesar de que cada versión cuenta con algunas características propias o pequeñas modificaciones, es evidente que todos siguen los mismos principios, por ello tomaremos como estándar al que se encuentra alojado en la aplicación de “Plato” que se encuentra disponible para celulares y para computadoras. Las características principales del software “Battle ship” alojado allí son:

- Tablero de juego de 10x10, con 6 naves por persona.
- Se juega por turnos, donde cada jugador coloca en el tablero sus barcos al empezar la partida.
- En sus turnos los jugadores pueden elegir donde caerá el proyectil (este no es lanzado, solo representa en que casilla busca atacar el usuario).
- El juego termina cuando uno de los dos jugadores elimina todas las embarcaciones del rival.
- Realizado principalmente en el lenguaje de programación conocido como “JavaScript”

En síntesis, lo anterior mencionado es el estándar de cómo realizar un juego de batallas navales. El objetivo de este proyecto de software es diferenciarse de esto último mejorando las mecánicas de lanzamiento de los misiles implementando detección de impacto con las tropas del rival, cambiando las dimensiones del tablero para adaptarlo a nuestra visión, ofreciendo un

lavado de cara a la estética del juego con una temática inspirada en lo “medieval”, creando un menú y por supuesto realizándolo en lenguaje C.

2.0 Descripción del Producto de Software

Nombre del Producto de Software: Videojuego sobre batallas estratégicas entre dos jugadores que utiliza mecánicas de lanzamiento de proyectiles mediante cañones para eliminar las tropas del oponente (“Abbys Battle”).

El producto de software lleva por nombre “**Abyss Battle**” será elaborado en lenguaje C. Será un videojuego para dos jugadores, de formato local y para computadoras de bajos recursos, considerando cómo bajo recursos a aquellos ordenadores con a lo menos 4GB de memoria RAM, que cuenten con sistema operativo “Windows”. El proyecto estará ambientado en la época medieval en sus personajes, escenarios y mecánicas, enfocado en diversión a los dos jugadores, por ello y para ahorrar tiempo en el desarrollo, carecerá de una historia y tendrá como principal objetivo desarrollar e implementar nuevas mecánicas.

Los jugadores podrán disfrutar de la experiencia de jugar “Battle Ship” de una forma diferente, ya que en “**Abyss Battle**” al iniciar una partida los usuarios eligen donde colocar a sus soldados, pero al jugarse en un mismo dispositivo los jugadores pueden ver donde coloco a sus soldados el contrincante, a su vez la forma de acabar con los soldados del equipo rival será a través de un cañón, siendo el ganador aquella persona que elimine antes a todos los soldados del equipo rival.

Las mecánicas que serán implementadas en el cañón son las protagonistas del combate, siendo a su vez la principal innovación que se piensa agregar al producto de software, para que se diferencie de un “Battle Ship” convencional. El cañón tendrá una mecánica de tipo lanzamiento cargado, esta consiste en que el jugador podrá dirigir y cargar un proyectil desplazando el cursor del “ratón” en la pantalla, esto se realizará en su turno correspondiente, teniendo como meta principal intentar hacer que impacte dicho proyectil sobre una de las tropas

enemigas, sin embargo, solo se indicará la potencia con la que se está lanzando el proyectil mediante una barra ubicada en un lateral de la pantalla de juego, a su vez se debe mostrar en pantalla solo el principio de la trayectoria del tiro, esto para hacer más complicado y divertido el juego, ya que se requerirá de una habilidad mayor para ganar una partida. De esta forma el videojuego “**Abyss Battle**” está pensado como un producto que se basa en un “Battle Ship” Convencional, tomando algunas ideas básicas de este, pero al mismo tiempo implementando mejoras en la ejecución.

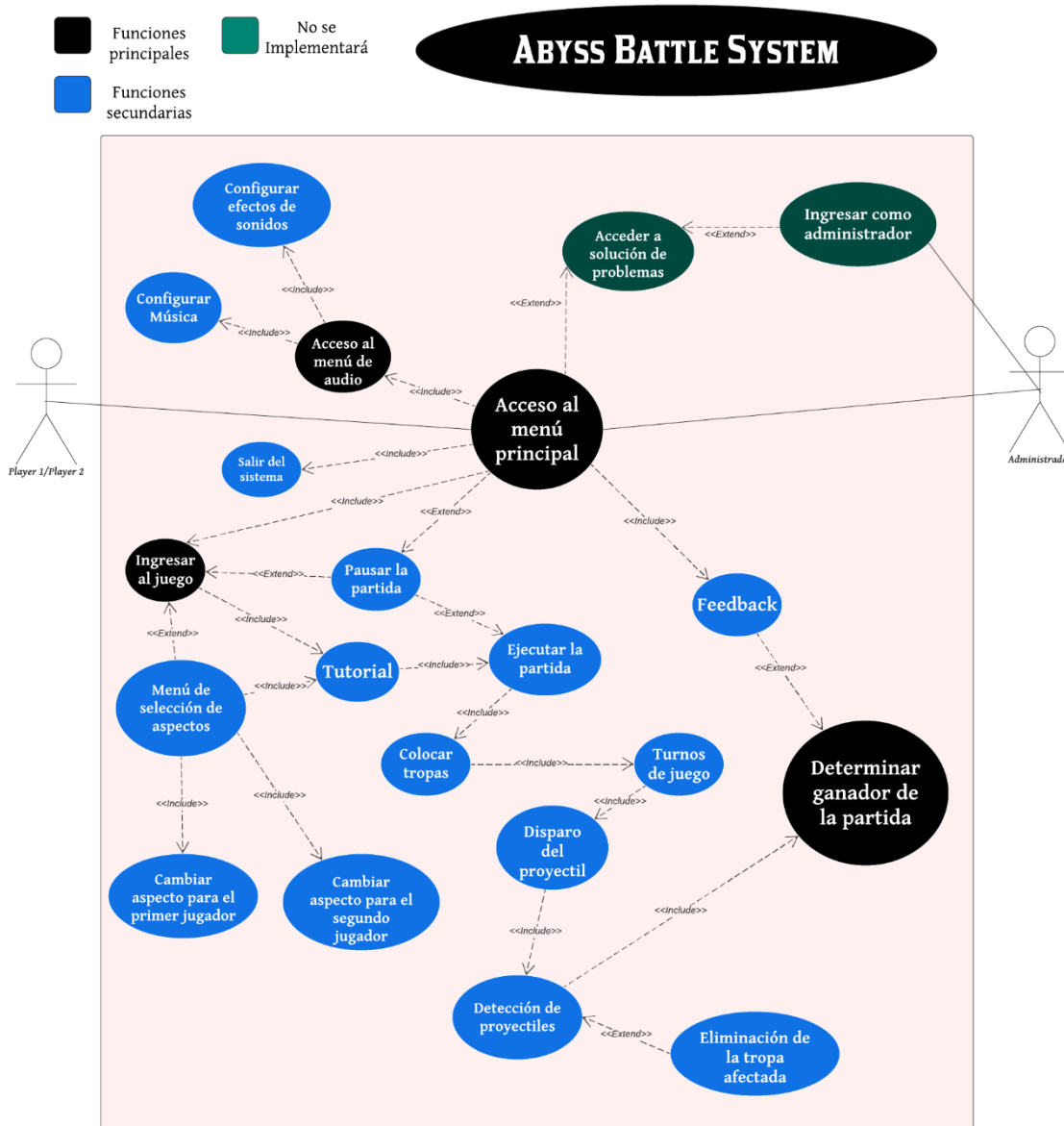
3.0 Objetivo General

Desarrollar un videojuego que simule un combate estratégico por turnos entre dos jugadores, en formato 2D, implementando como elemento principal la mecánica de lanzamiento y precisión con un cañón.

4.0 Objetivos Específicos del Sistema

1. Desarrollar una mecánica que permita calcular la trayectoria de un proyectil con cierta potencia, siendo este a su vez direccionado.
2. Optimizar el software para garantizar un rendimiento fluido en computadoras de escritorio y portátiles que cuenten con sistema operativo “Windows”.
3. Implementar un sistema de juego basado en turnos limitados por un tiempo, que permita detectar el cambio de turno entre usuarios.
4. Utilizar un diseño de interfaces intuitivo para el sistema, que permita que los usuarios se adapten y aprendan a utilizar el videojuego.
5. Crear un algoritmo de detección de colisiones contra objetos.
6. Incorporar en el software gráficos en 2D, animaciones, efectos de sonido y un “soundtrack” de uso libre.

5.0 Diagrama de Casos de Uso



6.0 Descripción de Tipos de Usuarios

6.1 Usuario Principal (Jugador)

6.1.1 Descripción: El jugador es la persona que interactúa directamente con el software, ya sea controlando el cañón y lanzando proyectiles durante las partidas o configurando opciones dentro del juego.

6.1.2 Responsabilidades:

- Jugar partidas contra otro jugador.
- Configurar opciones del juego, como ajustes de audio.

6.1.3 Características:

- Tiene habilidades y conocimientos básicos para jugar videojuegos.
- Puede variar en experiencia desde principiantes hasta jugadores más experimentados.

6.1.4 Objetivos:

- Disfrutar de la experiencia de juego.
- Ganar partidas contra otros jugadores.
- Explorar y descubrir las diferentes mecánicas y estrategias del juego.

6.2 Administrador

6.2.1 Descripción: El administrador es responsable de gestionar aspectos técnicos y de gestión del juego.

6.2.2 Responsabilidades:

- Realizar actualizaciones y mantenimiento del juego.
- Solucionar problemas técnicos o de rendimiento que presenten los usuarios principales.

6.2.3 Características:

- Puede ser un desarrollador del juego o un administrador de sistemas.
- Tiene conocimientos técnicos sobre programación y administración de servidores y resolución de problemas.

6.2.4 Objetivos:

- Mantener el juego funcionando correctamente.
- Proporcionar soporte técnico a los jugadores en caso de problemas.

7.0 Interfaces de Usuario

7.1 Interfaces de menú



Figura 1: Interfaces de los menús del sistema.



Figura 2: Flujo de las interfaces que el usuario visualiza al iniciar una partida

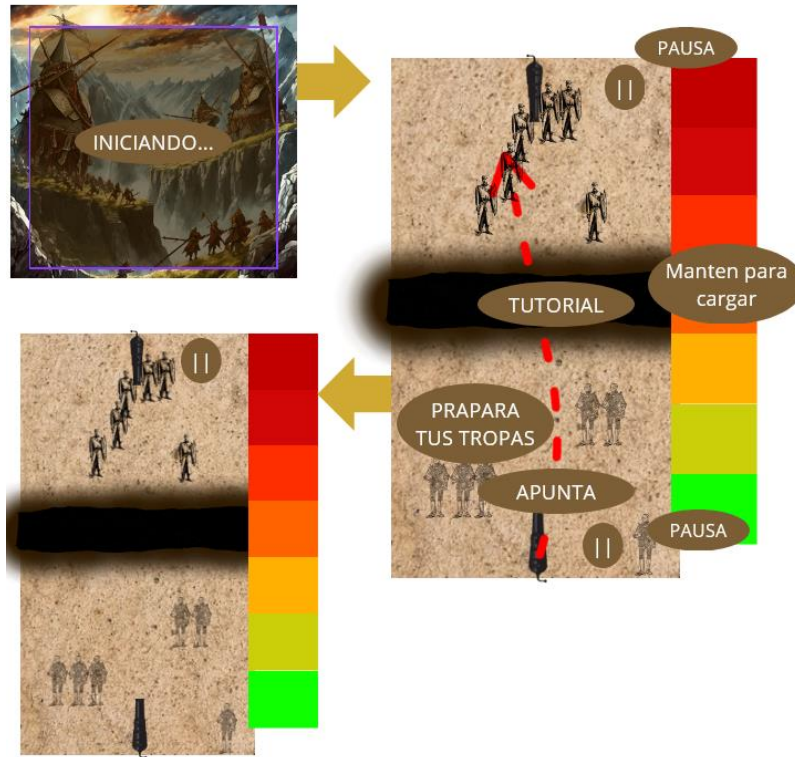


Figura 3: Interfaces que visualiza el usuario durante la ejecución de una partida normal en le software.

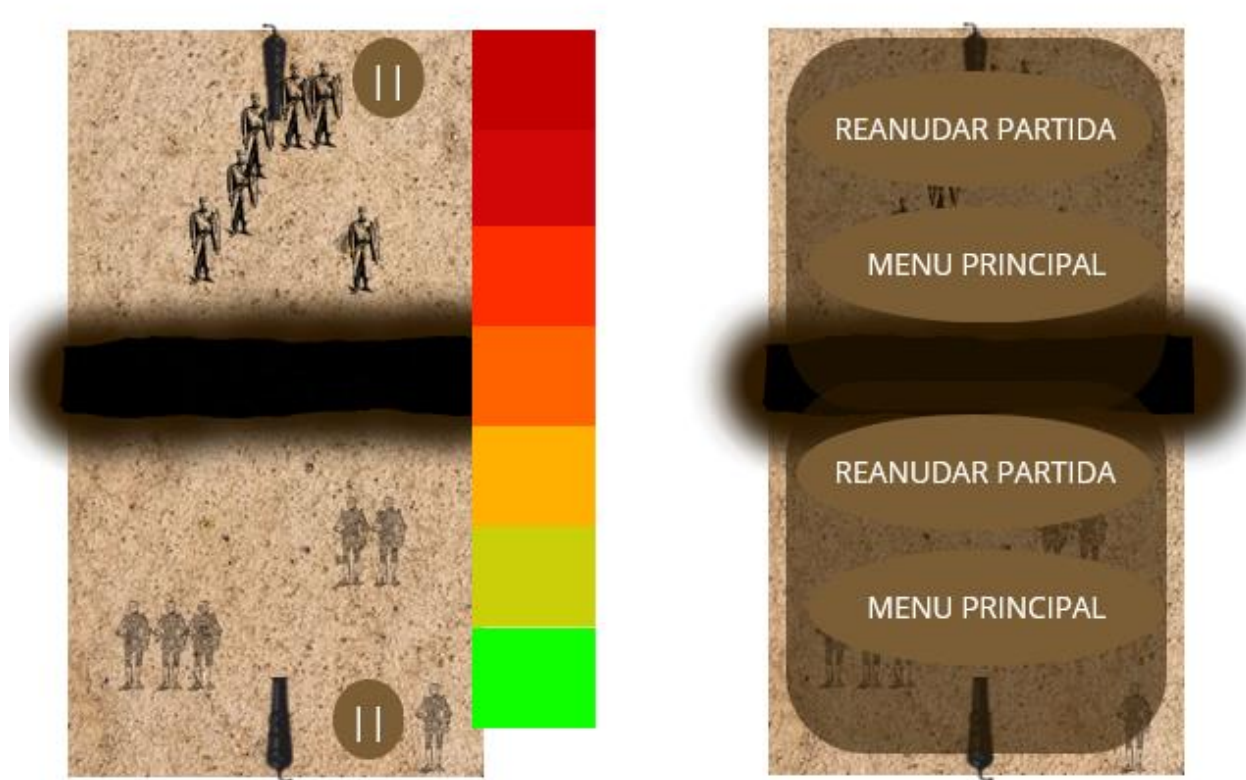


Figura 4: Interfaz del menú de pausa durante la ejecución de una partida.

8.0 Definición del estándar de codificación

De forma general se utilizará el compilador de “Dev C++” en su versión 5.11 para el desarrollo de la lógica del producto de software, por otro lado, se usará la biblioteca externa `freeglut.h` para habilitar funciones de OpenGL y poder añadir ventanas y gráficos.

para generar los gráficos del videojuego.

Para desarrollar un proyecto de este tamaño, será necesario llevar una documentación adecuada de los archivos importantes para su elaboración, además se debe establecer un estándar de codificación para promover el mutuo entendimiento entre integrantes y así optimizar el tiempo de codificación, ya que de no realizarse se extenderá de forma innecesaria el desarrollo del producto de software.

Se consideró una alternativa viable para establecer una forma legible de realizar el código, al "Estándar de Codificación de GNU", un proyecto de software libre iniciado en enero de 1984 a cargo de Richard Stallman, por ello cuenta con mucha documentación sobre el mismo. GNU consiste en un amplio conjunto de programas, con aplicaciones, librerías, herramientas para el desarrollo y juegos; con el tiempo se volvió tan grande que se tuvo que implementar reglas y recomendaciones para mantener un código legible y de alta calidad.

Con base al estándar anteriormente mencionado se busca establecer pautas claras y específicas para el proyecto "Abyss Battle", tomando los aspectos que se consideran más relevantes de GNU. A continuación, se proporciona el estándar que se usará para realizar este proyecto de software.

Estándar de Codificación para el Proyecto "Abyss Battle"

1. Legibilidad y Claridad:

- El código debe ser fácil de leer y entender para cualquier persona que lo revise.
- Utiliza nombres descriptivos y significativos para las variables, funciones y archivos.

2. Consistencia:

- Sigue un estilo de codificación consistente en todo el proyecto.
- Utiliza el mismo estilo de nombrado de variables, funciones y archivos en todo el código.
- Mantén una estructura de directorios consistente para los archivos del proyecto.

3. Portabilidad:

- Escribe código que sea portable y pueda ejecutarse en diferentes sistemas y compiladores.
- Evitar características específicas de una plataforma o compilador en tu código, salvo que sea necesario.

4. Uso de Estándares Aceptados:

- Sigue las convenciones y estándares de la industria para el nombrado de variables, funciones y archivos.
- Utiliza las bibliotecas estándar de C y SDL de manera apropiada y conforme a sus respectivas convenciones de codificación.

5. Documentación y Comentarios:

- Documenta el propósito y funcionamiento de cada función y módulo en comentarios claros y concisos.

- Utiliza comentarios para explicar partes importantes del código, incluyendo algoritmos complejos, decisiones de diseño y aspectos no obvios.

El "Estándar de Codificación de GNU" no es un documento único, sino más bien un conjunto de convenciones y prácticas de codificación promovidas por el Proyecto GNU. Revisar referencias relacionadas al final del documento.

9.0 Descripción de requerimientos funcionales y no funcionales

Requerimientos funcionales (RF) - Tabla 1	
Nombre	Descripción
RF-001. Menú principal	El sistema deberá contar con un menú principal de opciones que se le muestre en pantalla al usuario al iniciar la ejecución del software. Dicho menú tendrá las opciones: <ol style="list-style-type: none"> 1. Jugar 2. Configurar audio 3. Salir
RF-002. Menú de audio	Eligiendo la opción de “Configurar audio” en el menú principal que el sistema permite a los usuarios ingresar al “menú de audio” este servirá para ajustar las configuraciones de sonido del juego.
RF-003. Configurar música	En el menú de audio existe una opción que permite subir, bajar o silenciar el volumen de la música del sistema.
RF-004. Configurar efectos de sonido	En el menú de audio existe una opción que permite subir, bajar o silenciar el volumen de los efectos de sonido del software.
RF-005. Ingresar al juego	Los usuarios podrán ingresar a la ejecución de una partida del juego usando la opción de “jugar” en el menú principal o por el contrario deberán ingresar la tecla “Enter” de su ordenador de mesa o portátil en el mismo menú. El sistema redirige a los usuarios a la pantalla de juego.
RF-006. Salir del sistema	El sistema permite que el usuario pueda salir de este último usando el botón de “salir” ubicado en el menú principal.
RF-007. Tutorial	Al iniciar la ejecución de la partida mediante el botón “jugar”, independientemente de si se eligieron aspectos o no para los personajes, el software despliega en pantalla del usuario un tutorial que le explica cómo funcionan las principales mecánicas del juego.
RF-008. Objetivo de destrucción	En la pantalla de tutorial se deberá establecer al o los usuarios que como objetivo principal del juego se debe buscar la destrucción de objetos, es decir, la eliminación de objetivos específicos (las tropas enemigas) mediante los lanzamientos de proyectiles.
RF-009. Menú de selección de aspecto.	Después de seleccionar la opción “jugar” en el menú principal, pero antes de iniciar una partida, el sistema mostrará en pantalla una ventana que pregunta si se desea ingresar al menú de selección de aspecto. Si el usuario selecciona la opción “sí” se mostrará en pantalla dicho menú para que los jugadores pueden elegir el aspecto físico que tendrán sus tropas en el tablero de juego. En dicho menú estarán las siguientes opciones: <ol style="list-style-type: none"> 1. Cambiar aspectos del jugador 1. 2. Cambiar aspectos del jugador 2. 3. Aceptar.

	Al elegir la opción de “Aceptar” el menú se cerrará y se terminará la selección de aspecto para sus personajes.
RF-010. Detectar la selección de aspectos.	Cuando se selecciona un aspecto por parte del usuario el sistema detecta cuál de los jugadores ha seleccionado el aspecto de sus tropas, deshabilitando este aspecto para el jugador contrario al que selecciono.
RF-011. Cambiar el aspecto para el primer o segundo jugador	Se selecciona la opción “Cambiar aspectos del jugador 1” o en su defecto “Cambiar aspectos del jugador 2”. El software despliega en pantalla para los usuarios una galería con todos los aspectos disponibles, el usuario puede desplazar de izquierda a derecha en esta galería dando “click” con el cursor del “ratón” a unos botones ubicados a los lados de la pantalla (estos tendrán flechas indicando la dirección a la que mueven la galería). Cuando tenga seleccionado el aspecto que desee y al darle con el cursor a la opción de “aceptar” centrada en la parte inferior, se habrá completado el cambio de aspecto y se regresará al “menú de selección de aspecto”.
RF-012. Interacción multijugador local.	Al iniciar la ejecución de una partida del juego, el software permite a los jugadores interactuar entre ellos en un entorno multijugador local de 1v1, asignándole turnos de acción al jugador uno y al jugador dos, de forma que el usuario que maneja el sistema se convertirá en un turno en el primer jugador, y en el siguiente será el segundo jugador.
RF-013. Tablero de juego	El software dentro de la escena designada para la ejecución de las partidas (donde los usuarios pueden jugar) cuenta con un tablero de juego de 60 casillas de largo con 30 casillas de ancho que mostrará en la pantalla del usuario. Este tablero deberá dividirse a la mitad por una franja de casillas donde ninguno de los dos jugadores no permitirá el despliegue de tropas, esta designada para ser el abismo que divide el tablero, que da nombre al juego.
RF-014. Colocar tropas	Los usuarios durante la ejecución de la partida pueden elegir donde colocar a sus 12 soldados en las casillas validas del tablero de juego. Esta acción se realizará en dos turnos de un minuto de duración cada uno, donde cada jugador tendrá su oportunidad de colocar a sus tropas.
RF-015. Potencia de proyectiles	Se implementa una mecánica, durante la ejecución de una partida de juego, que permita cargar la potencia de un proyectil al arrastrar hacia atrás el cursor del “ratón” de la computadora, con una precisión de fuerza basada en la potencia que el usuario elija.
RF-016. Trayectoria de proyectiles	Se calcula de forma interna la trayectoria que llevará el proyectil que será lanzado, mostrando en la pantalla para el jugador en turno el inicio de esta trayectoria previamente calculada, evitando que se le muestre el total de la trayectoria al usuario.

RF-017. Turnos de los jugadores	Una vez que cada uno de los jugadores hayan colocado sus tropas, durante la ejecución de las partidas se contará con un temporizador de 20 segundos por turno de juego para que los usuarios puedan calcular y ejecutar su tiro. Al terminar el tiempo designado, se impedirá la acción de dicho jugador y se habilitará un “cooldown” de 5 segundos para que el siguiente jugador tome el dispositivo, después de este tiempo se habilitará el siguiente turno para el otro jugador.
RF-018. Detección de proyectiles	El software detecta cuando un proyectil impacta en una tropa enemiga durante la ejecución del turno de uno de los jugadores, eliminando el Sprite de dicha tropa de la pantalla y reduciendo el número de tropas del jugador que perdió a su guerrero en uno. Al detectarse un impacto de un proyectil contra una tropa enemiga, el sistema añade 10 segundos al temporizador del turno y permite otro turno. En caso no de que no sucede ningún impacto, el tiempo del turno se acaba automáticamente y se pasa al siguiente turno habilitando previamente un “cooldown” de 5 segundos antes del siguiente turno, dicho tiempo se muestra en pantalla para que los usuarios estén enterados.
RF-019. Determinar ganador	Después del lanzamiento de cualquiera de los jugadores y de que el sistema haya detectado la colisión o no colisión del proyectil, también analiza y determina cuando uno de los dos jugadores se queda sin tropas en el tablero, en caso de que se cumpla esta condición entonces la partida termina en ese momento. Al terminarse la partida en pantalla se despliega cuál de los dos usuarios fue el ganador y se devuelve a la pantalla de menú principal.
RF-020. Feedback	Al terminar la partida, se retroalimentará a los dos jugadores sobre el número de lanzamientos, derribos y tropas restantes que tuvo cada uno. Esta retroalimentación se imprimirá en pantalla al usuario en forma de ventana y se deberá seleccionar la opción de “Aceptar” centrada en la parte inferior para terminar de forma oficial la ejecución de una partida del juego.
RF-021. Pausar la partida	El software cuenta con un botón de pausa durante la escena donde se desarrolla la partida entre los jugadores, deberá ubicarse en la esquina superior izquierda de la pantalla y permite desplegar una ventana opciones. Entre las opciones se encuentra “salir al menú principal” o reanudar la partida. Al reanudarse la partida se activa un tiempo de espera de 3 segundos antes de que cualquier de los dos usuarios pueda realizar una acción en su turno. Al seleccionar “Salir al menú principal” el sistema devuelve al usuario al menú principal.
RF-022. Solución de problemas	Se cuenta con una opción en el menú principal para contactar a los administradores para la solución de problemas bien para ingresar como un administrador. Si no es administrador, se deberá seleccionar la opción de ayuda y luego se desplegará en pantalla,

	para los usuarios, la información de contacto. Para esta opción será necesario que exista una conexión a internet.
RF-023. Ingresar como administrador	Los administradores pueden ingresar al sistema para favorecer a la resolución de problemas de los usuarios. Para ello necesitarán contar con conexión a internet, ingresar a la opción administradores del menú principal, posteriormente seleccionar “soy administrador” e ingresar su usuario y contraseña.

Nota: Los colores que se encuentran en la tabla 1 y tabla 2 están basados en el método MoSCoW. Estos explican la importancia de cada uno de los requerimientos basándose en los resultados de la matriz de requerimientos y después en el criterio del equipo. Color verde representa aquellos que consideramos esenciales para el sistema. Color amarillo representa los requerimientos que deberían agregarse para mejorar la calidad, pero no son esenciales. Azul aquellos que pueden agregarse sin mucha dificultad. Rojo aquellos que fueron descartados por la matriz de requerimientos ya que no concuerdan con nuestros objetivos.

Requerimientos no funcionales (RNF) - Tabla 2	
Nombre	Descripción
RNF-001	El sistema debe estar optimizado para poder utilizarse en computadores que cuenten con un sistema operativo Windows 8 en adelante.
RNF-002	El juego debe ser capaz de ejecutarse de manera fluida y sin retrasos en el dispositivo, manteniendo una tasa de frames por segundo (FPS) adecuada.
RNF-003	Se debe optimizar para que funcione de forma fluida en computadoras con mínimo 4 GB de RAM en adelante.
RNF-004	El software debe contar con un diseño de interfaces intuitivas y minimalista que permita a los usuarios navegar fácilmente entre sus opciones y que además ayude a tener un rápido aprendizaje de cómo utilizarlo.
RNF-005	Los controles del juego responden de manera precisa y segura a las acciones del jugador, reaccionando a las acciones que éste decida
RNF-006	Debe estar optimizado para su ejecución de forma local en ordenadores de sobre mesa y portátiles (laptops), es decir, no se debe requerir que exista una conexión a internet para poder utilizar el sistema.
RNF-007	El sistema implementa gráficos y animaciones retro en 2D que permitan mayor inmersión para los jugadores, teniendo como principal objetivo reforzar la temática medieval del juego.
RNF-008	Cuenta con efectos de sonidos y música ambiental (“soundtrack”). La música ambiental debe cambiar por cada escenario, es decir, será diferente para los menús con respecto a la ejecución de la partida.

10.0 Descripción de los casos de uso

Caso 1: Acceso al menú principal

Descripción del caso de uso

Actor(es)	Jugador 1 - 2, Administrador.
Descripción	Tanto el usuario que funge como los jugadores como el administrador al ejecutar el software se les mostrará en pantalla el menú principal.
Precondiciones	El sistema debe estar instalado en un dispositivo que cuente con el sistema operativo Windows. El usuario debe ejecutar el sistema en su computadora.
Poscondiciones	El usuario visualiza el menú principal en su pantalla
Subcaso(s)	Ninguno

Actor		Sistema		
Paso	Acción	Paso	Acción	Excepción
1	El usuario ejecuta el sistema en su computadora	1	La aplicación muestra en la pantalla del usuario el menú principal del sistema, mientras inicia la ejecución del sonido de la aplicación. El menú principal despliega las siguientes opciones: Jugar Configurar audio Salir	Ninguna

Flujo de caso de uso

Excepciones: Este caso de uso no cuenta con ninguna excepción

Caso 2: Ingresar al juego

Descripción del caso de uso

Actor(es)	Jugador 1 - 2
Descripción	El usuario que funge como los jugadores ingresa a la ejecución de una partida del juego utilizando como medio el menú principal.
Precondiciones	<ul style="list-style-type: none"> • Debe haber ejecutado previamente la aplicación en su computadora. • Debe haberle dado “click” a la opción de jugar en el menú principal, o en su defecto haber “clickeado” la tecla “enter”
Poscondiciones	El sistema despliega dos opciones en la pantalla del jugador antes de iniciar la partida.
Subcaso(s)	Selección de aspecto, tutorial, pausar partida, turnos de juego.

Actor		Sistema		
Paso	Acción	Paso	Acción	Excepción
1	El usuario selecciona la opción de “jugar” en el menú principal.	1	El sistema cambia de pantalla y muestra ahora al usuario las siguientes opciones: 1. Cambiar el aspecto de las tropas 2. Iniciar partida	E1
2	Se selecciona “Iniciar partida en el menú”	2	El sistema empieza a cargar, una vez termine de cargar el escenario despliega en pantalla en tablero de juego y un tutorial.	E2

Flujo de caso de uso

Excepciones

Identificador	Nombre	Acción
---------------	--------	--------

E1	Forzar salir del sistema con alt+f4.	El sistema se cierra automáticamente y termina la ejecución.
E2	Se selecciona “Cambiar el aspecto de las tropas”	El sistema se dirige al subcaso “Selección de aspecto”.

Caso 3: Selección de aspecto

Descripción del caso de uso

Actor(es)	Jugador 1 - 2
Descripción	Los jugadores eligen por turnos el aspecto que quieren que tengan sus tropas durante la partida que jugaran.
Precondiciones	Deben haber elegido en el menú principal la opción de jugar. Deben haber elegido la opción de “cambiar el aspecto de las tropas”
Poscondiciones	Se cambia el aspecto de las tropas del jugador 1 y 2 para esta partida.
Subcaso(s)	Ninguno.

Actor		Sistema		
Paso	Acción	Paso	Acción	Excepción
1	El jugador 1 elige el aspecto de sus tropas dentro de las opciones mostradas en pantalla.	1	En el software se cambia el aspecto de las tropas para el jugador 1 en dicha partida.	
2	El jugador 2 elige el aspecto de sus tropas dentro de las opciones mostradas en pantalla.	2	En el software se cambia el aspecto de las tropas para el jugador 2 en dicha partida.	E1
3	Se selecciona la opción de aceptar.	3	El sistema sale de este menú y redirige al usuario (jugador 1-2) a la partida.	

Flujo de caso de uso

Excepciones

Identificador	Nombre	Acción
E1	Eligió el mismo aspecto que el jugador 1	El sistema le muestra en pantalla una ventana que diga “No puedes elegir el mismo aspecto que el jugador anterior”.

Caso 4: Tutorial

Descripción del caso de uso

Actor(es)	Jugador 1 - 2
Descripción	El sistema despliega en la pantalla del usuario una ventana dando la bienvenida al tutorial. Tiene como objetivo explicar a los jugadores las principales mecánicas del software durante una partida.
Precondiciones	Debe haber seleccionado la opción de jugar en el menú principal. Debe haber seleccionado la opción de iniciar partida o en su defecto haber pasado por la selección de aspecto y después haberle dado a iniciar.
Poscondiciones	Se desea suerte a los jugadores e inicia la partida.
Subcaso	Ninguno.

Actor		Sistema		
Paso	Acción	Paso	Acción	Excepción
1	El usuario inicia la partida	1	En el sistema se despliega una ventana dándole la bienvenida, preguntando si desea ver el tutorial.	
2	Se le indica al sistema que sí se desea ver el tutorial.	2	Se despliega una pantalla de información con una explicación básica sobre el funcionamiento de las mecánicas del juego.	E1
3	Se presiona el botón de “aceptar” con el mouse o se presiona la tecla “enter”.	3	Se termina el y se cierra la ventana de información.	

Flujo de caso de uso

Excepciones

Identificador	Nombre	Acción
---------------	--------	--------

E1	Se le indica al sistema que no se desea ver el tutorial	El sistema cierra la ventana del tutorial y pasa directamente a la partida.
----	---	---

Caso 5: Pausar la partida

Descripción del caso de uso

Actor(es)	Jugador 1 - 2, Administrador.
Descripción	Los jugadores dan click en el botón de “pausar” durante la ejecución de la partida.
Precondiciones	Se debe haber iniciado una partida en el sistema.
Poscondiciones	La partida se detiene, ahora los usuarios pueden elegir si volver al menú principal o reanudar la partida.
Subcaso	ninguno

Actor		Sistema		
Paso	Acción	Paso	Acción	Excepción
1	Se da “click” al botón de “pausar partida” durante la ejecución del juego.	1	El sistema despliega en pantalla una ventana con dos opciones: 1. Regresar al menú principal 2. Reanudar partida	E1
2	El usuario selecciona “Regresar al menú principal”	2	Se despliega una ventana preguntando si está seguro de que quiere realizar dicha acción.	E2
3	Se confirma la acción	3	El sistema devuelve al caso de uso 1.	

Flujo de caso de uso

Excepciones

Identificador	Nombre	Acción
E1	Selección de “reanudar partida”.	Se devuelve a la pantalla de la partida justo donde se quedó.
E2	Se cancela la acción	Se devuelve al paso 1 del caso de uso.

Caso 6: Turnos de juego

Descripción del caso de uso

Actor(es)	Jugador 1 - 2, Administrador.
Descripción	Al iniciar la partida se muestra en pantalla de quien de los jugadores es el turno y se habilitan las acciones para este último.
Precondiciones	Deben haber iniciado la partida. Deben haber pasado por el caso de uso del tutorial.
Poscondiciones	El jugador decide que acciones realizar en su turno.
Subcaso(s)	Disparo de proyectil, detección de proyectiles, colocar tropas.

Actor		Sistema		
Paso	Acción	Paso	Acción	Excepción
1	Se inicia la partida.	1	Se muestra en pantalla de cuál de los jugadores es el turno en este momento. Se inicia el temporizador de tiempo de dicho turno.	
2	El jugador dispara en su turno, se acierta el disparo a uno de los rivales.	2	Se elimina del tablero a la tropa enemiga donde impacto el proyectil. Se activa el tiempo de enfriamiento y se le otorgan 10 segundos extra al jugador en turno para volver a lanzar.	E1
3	Se termina el turno.	3	Se cambia el turno al otro jugador y se muestra en pantalla quien va a jugar. Se vuelve a iniciar el temporizador de tiempo y se regresa al paso 2.	





Flujo de caso de uso







Excepciones

Identificador	Nombre	Acción
E1	Fallo del disparo	El sistema salta al paso 3 del caso.






11.0 Matriz de Requerimientos




Requerimientos funcionales:	Objetivos específicos:					
	1	2	3	4	5	6
<p>RF-001.</p> <p>Menú principal</p> <p>El sistema deberá contar con un menú principal de opciones que se le muestre en pantalla al usuario al iniciar la ejecución del software. Dicho menú tendrá las opciones:</p> <ol style="list-style-type: none"> 1. Jugar 2. Configurar audio 3. Salir 				<input checked="" type="checkbox"/>		
<p>RF-003.</p> <p>Menú de audio</p> <p>Eligiendo la opción de “Configurar audio” en el menú principal que el sistema permite a los usuarios ingresar al “menú de audio” este servirá para ajustar las configuraciones de sonido del juego.</p>					<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<p>RF-004.</p> <p>Configurar música</p> <p>En el menú de audio existe una opción que permite subir, bajar o silenciar el volumen de la música del sistema.</p>						<input checked="" type="checkbox"/>
<p>RF-005</p> <p>Configurar efectos de sonido</p> <p>En el menú de audio existe una opción que permite subir, bajar o silenciar el volumen de los efectos de sonido del software.</p>						<input checked="" type="checkbox"/>
<p>RF-006.</p> <p>Ingresar al juego</p> <p>Los usuarios podrán ingresar a la ejecución de una partida del juego usando la opción de “jugar” en el menú principal o por el contrario deberán ingresar la tecla “Enter” de su</p>				<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>

ordenador de mesa o portátil en el mismo menú. El sistema redirige a los usuarios a la pantalla de juego.						
RF-007. Salir del sistema El sistema permite que el usuario pueda salir de este último usando el botón de “salir” ubicado en el menú principal.						
RF-008. Tutorial Al iniciar la ejecución de la partida mediante el botón “jugar”, independientemente de si se eligieron aspectos o no para los personajes, el software despliega en pantalla del usuario un tutorial que le explica cómo funcionan las principales mecánicas del juego.						
RF-009. Objetivo de destrucción En la pantalla de tutorial se deberá establecer al o los usuarios que como objetivo principal del juego se debe buscar la destrucción de objetos, es decir, la eliminación de objetivos específicos (las tropas enemigas) mediante los lanzamientos de proyectiles.						
RF-010. Menú de selección de aspecto. Después de seleccionar la opción “jugar” en el menú principal, pero antes de iniciar una partida, el sistema mostrará en pantalla una ventana que pregunta si se desea ingresar al menú de selección de aspecto. Si el usuario selecciona la opción “sí” se mostrará en pantalla dicho menú para que los jugadores pueden elegir el aspecto físico que tendrán sus tropas en el tablero de juego. En dicho menú estarán las siguientes opciones: Cambiar aspectos del jugador 1. Cambiar aspectos del jugador 2. Aceptar.						

Al elegir la opción de “Aceptar” el menú se cerrará y se terminará la selección de aspecto para sus personajes.						
RF-011. Detectar la selección de aspectos. Cuando se selecciona un aspecto por parte del usuario el sistema detecta cuál de los jugadores ha seleccionado el aspecto de sus tropas, deshabilitando este aspecto para el jugador contrario al que selecciono.						
RF-012. Cambiar el aspecto para el primer o segundo jugador Se selecciona la opción “Cambiar aspectos del jugador 1” o en su defecto “Cambiar aspectos del jugador 2”. El software despliega en pantalla para los usuarios una galería con todos los aspectos disponibles, el usuario puede desplazar de izquierda a derecha en esta galería dando “click” con el cursor del “ratón” a unos botones ubicados a los lados de la pantalla (estos tendrán flechas indicando la dirección a la que mueven la galería). Cuando tenga seleccionado el aspecto que desee y al darle con el cursor a la opción de “aceptar” centrada en la parte inferior, se habrá completado el cambio de aspecto y se regresará al “menú de selección de aspecto”.						
RF-013. Interacción multijugador local. Al iniciar la ejecución de una partida del juego, el software permite a los jugadores interactuar entre ellos en un entorno multijugador local de 1v1, asignándole turnos de acción al jugador uno y al jugador dos, de forma que el usuario que maneja el sistema se convertirá en un turno en el primer jugador, y en el siguiente será el segundo jugador.						
RF-014. Tablero de juego						

El software dentro de la escena designada para la ejecución de las partidas (donde los usuarios pueden jugar) cuenta con un tablero de juego de 60 casillas de largo con 30 casillas de ancho que mostrará en la pantalla del usuario. Este tablero deberá dividirse a la mitad por una franja de casillas donde ninguno de los dos jugadores no permitirá el despliegue de tropas, esta designada para ser el abismo que divide el tablero, que da nombre al juego.						
<p>RF-015.</p> <p>Colocar tropas</p> <p>Los usuarios durante la ejecución de la partida pueden elegir donde colocar a sus 12 soldados en las casillas validas del tablero de juego. Esta acción se realizará en dos turnos de un minuto de duración cada uno, donde cada jugador tendrá su oportunidad de colocar a sus tropas.</p>			✓			
<p>RF-016.</p> <p>Potencia de proyectiles</p> <p>Se implementa una mecánica, durante la ejecución de una partida de juego, que permita cargar la potencia de un proyectil al arrastrar hacia atrás el cursor del “ratón” de la computadora, con una precisión de fuerza basada en la potencia que el usuario elija.</p>	✓	✓				
<p>RF-017.</p> <p>Trayectoria de proyectiles</p> <p>Se calcula de forma interna la trayectoria que llevará el proyectil que será lanzado, mostrando en la pantalla para el jugador en turno el inicio de esta trayectoria previamente calculada, evitando que se le muestre el total de la trayectoria al usuario.</p>	✓	✓				
<p>RF-018.</p> <p>Turnos de los jugadores</p> <p>Una vez que cada uno de los jugadores hayan colocado sus tropas, durante la ejecución de las partidas se contará con un temporizador de 20 segundos por turno de juego para que los usuarios puedan calcular y ejecutar su tiro. Al</p>			✓			

terminar el tiempo designado, se impedirá la acción de dicho jugador y se habilitará un “cooldown” de 5 segundos para que el siguiente jugador tome el dispositivo, después de este tiempo se habilitará el siguiente turno para el otro jugador.						
<p>RF-019.</p> <p>Detección de proyectiles</p> <p>El software detecta cuando un proyectil impacta en una tropa enemiga durante la ejecución del turno de uno de los jugadores, eliminando el Sprite de dicha tropa de la pantalla y reduciendo el número de tropas del jugador que perdió a su guerrero en uno. Al detectarse un impacto de un proyectil contra una tropa enemiga, el sistema añade 10 segundos al temporizador del turno y permite otro turno. En caso no de que no sucede ningún impacto, el tiempo del turno se acaba automáticamente y se pasa al siguiente turno habilitando previamente un “cooldown” de 5 segundos antes del siguiente turno, dicho tiempo se muestra en pantalla para que los usuarios estén enterados.</p>						
<p>RF-020.</p> <p>Determinar ganador</p> <p>Después del lanzamiento de cualquiera de los jugadores y de que el sistema haya detectado la colisión o no colisión del proyectil, también analiza y determina cuando uno de los dos jugadores se queda sin tropas en el tablero, en caso de que se cumpla esta condición entonces la partida termina en ese momento. Al terminarse la partida en pantalla se despliega cuál de los dos usuarios fue el ganador y se devuelve a la pantalla de menú principal.</p>						
<p>RF-021.</p> <p>Feedback</p> <p>Al terminar la partida, se retroalimentará a los dos jugadores sobre el número de lanzamientos, derribos y tropas restantes que tuvo cada uno. Esta retroalimentación se imprimirá en pantalla al usuario en forma de ventana y se deberá seleccionar la opción de “Aceptar” centrada en la parte inferior para terminar de forma oficial la ejecución de una partida del juego.</p>						

<p>RF-022.</p> <p>Pausar la partida</p> <p>El software cuenta con un botón de pausa durante la escena donde se desarrolla la partida entre los jugadores, deberá ubicarse en la esquina superior izquierda de la pantalla y permite desplegar una ventana opciones. Entre las opciones se encuentra “salir al menú principal” o reanudar la partida. Al reanudarse la partida se activa un tiempo de espera de 3 segundos antes de que cualquier de los dos usuarios pueda realizar una acción en su turno. Al seleccionar “Salir al menú principal” el sistema devuelve al usuario al menú principal.</p>						
<p>sRF-023.</p> <p>Solución de problemas</p> <p>Se cuenta con una opción en el menú principal para contactar a los administradores para la solución de problemas bien para ingresar como un administrador. Si no es administrador, se deberá seleccionar la opción de ayuda y luego se desplegará en pantalla, para los usuarios, la información de contacto. Para esta opción será necesario que exista una conexión a internet.</p>						
<p>RF-024.</p> <p>Ingresar como administrador</p> <p>Los administradores pueden ingresar al sistema para favorecer a la resolución de problemas de los usuarios. Para ello necesitarán contar con conexión a internet, ingresar a la opción administradores del menú principal, posteriormente seleccionar “soy administrador” e ingresar su usuario y contraseña.</p>						

12.0 Proceso de Desarrollo

12.1 Herramientas utilizadas

12.1.1 Dev C++

Se utiliza este entorno de desarrollo integrado para programar la lógica del producto de software en lenguaje C, esto porque es el programa estándar con el que se trabajó durante el curso. Adicionalmente se usarán la biblioteca externa freeglut.h para habilitar funciones de OpenGL y poder añadir ventanas y gráficos.

12.1.3 Figma

Se utiliza Figma para la elaboración de las interfaces de usuario de baja y alta fidelidad para los módulos del producto de software.

12.1.4 GitHub – GitKraken

Se utilizan estos programas para llevar un control de versiones, tanto del código como de la documentación asociada a este. Se realiza en un repositorio privado y es parte de la organización interna del equipo.

12.2 Organización

La organización se realizará de manera informal primero para garantizar mayor fluidez y rapidez, y después se organizarán las tareas a realizar más formalmente.

12.2.1 WhatsApp – Discord

Los primeros canales de comunicación del equipo y los más informales, pero son los más rápidos al transmitir ideas.

12.2.2 To do List

Almacenado en el repositorio privado en de este proyecto en Github, se encuentra la “To do list” en la en el tablero Kanban. Nos sirve para registrar las tareas del desarrollo pendientes, las urgentes, identificar las que se encuentran en proceso, y visualizar las que ya fueron completadas con éxito.

12.3 Monitoreo

12.3.1 GitHub – GitKraken

Cómo se mencionó anteriormente, se utiliza Github y Gitkraken para llevar un control de versiones, tanto del código como de la documentación asociada a este. Además de las funciones anteriormente mencionadas, también permiten guardar los procesos y participaciones individuales en forma del número de “pull request” de cada integrante del equipo, de esta forma sirve como control de versiones.

12.4 Bitácoras

Se especificará una fecha y hora máxima para realizar las actividades correspondientes de la elaboración de los proyectos asignados a cada persona para evitar contratiempos. Esta información se podrá encontrar de manera específica para cada tarea pendiente en la “to do list” ubicada en el repositorio del proyecto.

12.5 Medición del trabajo grupal

El trabajo grupal y el desempeño del equipo se determinarán mediante reuniones diarias dirigidas por un SCRUM máster o en su defecto el líder de equipo, en estas reuniones se compartirán los avances de cada integrante, se expondrán dudas y se analizará cómo se progresó según la bitácora, si hay un problema se busca identificarlo y solucionarlo. Además, se

procurará tomar en cuenta opiniones o cualquier aporte dado para la implementación y mejora del proyecto por parte de un integrante.

12.6 Medición del trabajo individual

Haciendo uso de la bitácora y de la actividad registrada por persona en el repositorio del proyecto, se medirá el trabajo individual. A lo mencionado, se le añadirá el cumplimiento de las actividades no relacionadas con la codificación asignadas a cada integrante del equipo, analizando si se cumplieron en tiempo y forma con respecto a los tiempos establecidos en la bitácora.

13.0 Avances del Código

Funciones que permite visualizar los menús, tanto el principal como el secundario (dentro de una partida)

```

201
202 // Función para mostrar el menú principal
203 void show_main_menu() {
204     printf("MENU PRINCIPAL\n");
205     printf("1. Iniciar juego\n");
206     printf("2. Cerrar programa\n");
207     printf("Seleccione una opción: ");
208 }
209
210 // Función para mostrar el menú secundario
211 void show_submenu() {
212     printf("\nMENU SECUNDARIO\n");
213     printf("2. Reanudar partida\n");
214     printf("3. Regresar al menú principal\n");
215     printf("Seleccione una opción: ");
216 }
217

```

Función para mostrar una representación visual de una tropa en el tablero

```

// Función para mostrar una representación visual de una tropa en el tablero
void visualize_troop(char board[BOARD_HEIGHT][BOARD_WIDTH], int row, int col, Troop *troop) {
    char symbol = '#'; // Representa una tropa
    if (troop->vertical) {
        for (int i = 0; i < troop->size; i++) {
            board[row + i][col] = symbol;
        }
    } else {
        for (int i = 0; i < troop->size; i++) {
            board[row][col + i] = symbol;
        }
    }
    print_board(board, "ABYSS'S BATTLE");
}

```

Función que lee las tropas para acomodarlas en cada tablero

```
// Función para que el jugador coloque sus tropas
void place_troops_manually(char board[BOARD_HEIGHT][BOARD_WIDTH], Troop *troops) {
    for (int i = 0; i < NUM_TROOPS; i++) {
        printf("Coloca la tropa de tamaño %d (formato: fila columna orientación (V/H)): ", troops[i].size);
        int row, col;
        char orientation;
        scanf("%d %d %c", &row, &col, &orientation);
        row--; // Ajustamos al índice base 0
        col--; // Ajustamos al índice base 0
        orientation = toupper(orientation);
        if (!is_valid_coordinate(row, col) || (orientation != 'V' && orientation != 'H')) {
            printf("Coordenadas inválidas. Por favor, inténtalo de nuevo.\n");
            i--;
            continue;
        }
        troops[i].vertical = (orientation == 'V');
        if (!can_place_troop(board, row, col, &troops[i])) {
            printf("La tropa no cabe en esta posición o se superpone con otra. Por favor, inténtalo de nuevo.\n");
            i--;
            continue;
        }
        visualize_troop(board, row, col, &troops[i]);
        place_troop(board, row, col, &troops[i]);
    }
}
```

Función que coloca las tropas en el tablero

```
// Función para colocar una tropa en el tablero
void place_troop(char board[BOARD_HEIGHT][BOARD_WIDTH], int row, int col, Troop *troop) {
    if (troop->vertical) {
        for (int i = 0; i < troop->size; i++) {
            board[row + i][col] = '#'; // Representa una tropa
        }
    } else {
        for (int i = 0; i < troop->size; i++) {
            board[row][col + i] = '#'; // Representa una tropa
        }
    }
}
```

Función para que el jugador realice un ataque

```
// Función para que el jugador realice un ataque
void player_attack(char board[BOARD_HEIGHT][BOARD_WIDTH], char opponent_board[BOARD_HEIGHT][BOARD_WIDTH], Troop *troops, bool *extra_shot) {
    int row, col;
    printf("Introduce las coordenadas de tu ataque (fila columna): ");
    scanf("%d %d", &row, &col);
    row--; // Ajustamos al índice base 0
    col--; // Ajustamos al índice base 0

    if (!is_valid_coordinate(row, col)) {
        printf("Coordenadas inválidas. Por favor, inténtalo de nuevo.\n");
        player_attack(board, opponent_board, troops, extra_shot); // Pedir coordenadas nuevamente
        return;
    }

    if (opponent_board[row][col] == '#') {
        printf("¡Impacto!\n Puede volver a atacar \n");
        board[row][col] = 'X';
        opponent_board[row][col] = 'X';
        // Marcar la tropa como golpeada
        for (int i = 0; i < NUM_TROOPS; i++) {
            if (troops[i].vertical) {
                if (col >= col && col < col + troops[i].size && row >= row && row < row + troops[i].size) {
                    troops[i].hit[col - row] = true;
                    // Comprobar si la tropa ha sido eliminada
                    bool eliminated = true;
                    for (int j = 0; j < troops[i].size; j++) {
                        if (!troops[i].hit[j]) {
                            eliminated = false;
                            break;
                        }
                    }
                    if (eliminated) {
                        printf("Tropa eliminada.\n");
                    }
                }
            }
        }
    }
}
```

```

    }
    if (eliminated) {
        printf("Has eliminado una tropa enemiga!\n");
    }
    break;
}
} else {
    if (row >= row && row < row + troops[i].size && col >= col && col < col + troops[i].size) {
        troops[i].hit[row - col] = true;
        // Comprobar si la tropa ha sido eliminada
        bool eliminated = true;
        for (int j = 0; j < troops[i].size; j++) {
            if (!troops[i].hit[j]) {
                eliminated = false;
                break;
            }
        }
        if (eliminated) {
            printf("Has eliminado una tropa enemiga!\n");
        }
        break;
    }
}
}
*extra_shot = true; // Dar un disparo adicional
} else if (opponent_board[row][col] == 'X' || opponent_board[row][col] == 'O') {
    printf("Ya has atacado esta posición. Por favor, elige otra.\n");
    player_attack(board, opponent_board, troops, extra_shot); // Pedir coordenadas nuevamente
} else {
    printf("Fallo...\n");
    board[row][col] = 'O';
    opponent_board[row][col] = 'O';
}
}
}

```

Función que verifica si algún jugador ha ganado

```

// Función para verificar si un jugador ha ganado
bool check_win(char board[BOARD_HEIGHT][BOARD_WIDTH]) {
    for (int i = 0; i < BOARD_HEIGHT; i++) {
        for (int j = 0; j < BOARD_WIDTH; j++) {
            if (board[i][j] == '#') {
                return false;
            }
        }
    }
    return true;
}

```

Función Main principal

```
int main() {
    setlocale(LC_ALL, "");
    char player1_board[BOARD_HEIGHT][BOARD_WIDTH];
    char player2_board[BOARD_HEIGHT][BOARD_WIDTH];
    char player1_shots[BOARD_HEIGHT][BOARD_WIDTH];
    char player2_shots[BOARD_HEIGHT][BOARD_WIDTH];

    // Inicializar tableros
    init_board(player1_board);
    init_board(player2_board);
    init_board(player1_shots);
    init_board(player2_shots);

    // Definir las tropas de cada jugador
    Troop player1_troops[NUM_TROOPS] = {{5, true}, {4, true}, {3, true}, {3, true}, {2, true}};
    Troop player2_troops[NUM_TROOPS] = {{5, true}, {4, true}, {3, true}, {3, true}, {2, true}};

    int main_menu_option;
    do {
        show_main_menu();
        scanf("%d", &main_menu_option);
        switch (main_menu_option) {
            case 1: {
                // Colocar tropas manualmente
                printf("Jugador 1, coloca tus tropas:\n");
                place_troops_manually(player1_board, player1_troops);
                printf("Jugador 2, coloca tus tropas:\n");
                place_troops_manually(player2_board, player2_troops);

                // Juego
                bool game_over = false;
                bool player1_turn = true;
                bool player1_extra_shot = false;
                bool player2_extra_shot = false;
                while (!game_over) {
                    print_board(player1_board, "Tablero del Jugador 1:");
```

Bucle principal del juego

```
        place_troops_manually(player2_board, player2_troops);

        // Juego
        bool game_over = false;
        bool player1_turn = true;
        bool player1_extra_shot = false;
        bool player2_extra_shot = false;
        while (!game_over) {
            print_board(player1_board, "Tablero del Jugador 1:");
            print_board(player2_board, "Tablero del Jugador 2:");

            if (player1_turn) {
                printf("\nTurno del Jugador 1:\n");
                player_attack(player1_shots, player2_board, player2_troops, &player1_extra_shot);
                game_over = check_win(player2_board);
                if (!player1_extra_shot) {
                    player1_turn = false;
                }
                player1_extra_shot = false;
            } else {
                printf("\nTurno del Jugador 2:\n");
                player_attack(player2_shots, player1_board, player1_troops, &player2_extra_shot);
                game_over = check_win(player1_board);
                if (!player2_extra_shot) {
                    player1_turn = true;
                }
                player2_extra_shot = false;
            }
        }
    }
```

Visualizar el Menú secundario durante la partida

```
// Mostrar el menú secundario
if (!game_over) {
    show_submenu();
    int submenu_option;
    scanf("%d", &submenu_option);
    switch (submenu_option) {
        case 1:
            // Reiniciar partida
            init_board(player1_board);
            init_board(player2_board);
            init_board(player1_shots);
            init_board(player2_shots);
            place_troops_manually(player1_board, player1_troops);
            place_troops_manually(player2_board, player2_troops);
            game_over = false;
            player1_turn = true;
            break;
        case 2:
            // Reanudar partida (no hace nada)
            break;
        case 3:
            // Regresar al menú principal
            game_over = true;
            break;
        default:
            printf("Opción inválida. Por favor, seleccione nuevamente.\n");
            break;
    }
}
```

Cierre del programa

```
        break;
    }
}

printf("Juego terminado! ");
if (player1_turn) {
    printf("El Jugador 1 ha ganado!\n");
} else {
    printf("El Jugador 2 ha ganado!\n");
}
break;
}
case 2:
    printf("Hasta luego!\n ---CREDITOS--- \n TEAM CFORCE \n Alonzo Palacios Rodrigo Alonzo \n Cuevas Garcia Braulio Samuel \n Martinez Martin");
    return 0;
default:
    printf("Opción inválida. Por favor, seleccione nuevamente.\n");
    break;
}
} while (true);

return 0;
```

14.0 Reporte de evaluación basada en objetivos

Integrante	Participaciones individuales	Evidencias	Aportación total para el equipo
Alonzo Palacios Rodrigo	<ul style="list-style-type: none"> • Elaboración de la presentación de los avances del proyecto • Codificación • Descripción de requerimientos y casos de uso • Revisión y corrección del archivo anterior Redacción en el archivo	<ul style="list-style-type: none"> • Archivo de reporte de resultados. • Presentación del reporte de resultados. • Código main del programa. 	100%
Cuevas García Braulio Samuel	<ul style="list-style-type: none"> • Elaboración de la presentación de los avances del proyecto • Codificación • Descripción de requerimientos y casos de uso • Revisión y corrección del archivo anterior Redacción en el archivo	<ul style="list-style-type: none"> • Archivo de reporte de resultados. • Presentación del reporte de resultados. • Código main del programa. 	100%
Martínez Martínez José Pablo	<ul style="list-style-type: none"> • Elaboración de la presentación de los avances del proyecto • Codificación • Descripción de requerimientos y casos de uso • Revisión y corrección del archivo anterior Redacción en el archivo	<ul style="list-style-type: none"> • Archivo de reporte de resultados. • Presentación del reporte de resultados. • Código main del programa. 	100%
Moo Pan Jareth Jaziel			100%

	<ul style="list-style-type: none">• Elaboración de la presentación de los avances del proyecto• Codificación• Descripción de requerimientos y casos de uso• Revisión y corrección del archivo anterior Redacción en el archivo	<ul style="list-style-type: none">• Archivo de reporte de resultados• Presentación del reporte de resultados• Código main del programa	
--	---	--	--

Referencias

Comments (GNU Coding Standards). (s. f.).

https://www.gnu.org/prep/standards/html_node/Comments.html

El sistema operativo GNU y el movimiento del software libre. (s. f.).

<https://www.gnu.org/home.es.html>

GNU Coding Standards - GNU Project - Free Software Foundation. (s. f.).

<https://www.gnu.org/prep/standards/>

<https://www.platoapp.com>

Information for Maintainers of GNU Software. (s. f.).

<https://www.gnu.org/prep/maintain/maintain.html>

Names (GNU Coding Standards). (s. f.).

https://www.gnu.org/prep/standards/html_node/Names.html