Proyecto (Valor 30%)
"CATAN"

Objetivos

1. Desarrollar un programa en donde se apliquen conceptos de estructuras de datos por medio de la programación en el lenguaje C++, aplicando criterios de lógica y formas óptimas de codificación para dar soluciones a problemas de origen informático.
2. Aplicar los conocimientos adquiridos durante el curso, en conceptos como arreglos, listas, grafos, archivos, etc.

Instrucciones

1. Lea y comprenda cuidadosamente lo que se le solicite.
2. El proyecto será desarrollado en grupos de hasta 3 estudiantes.
3. El proyecto debe ser desarrollado utilizando el Paradigma de Orientación a Objetos, aplicando el lenguaje C++, mediante la herramienta IDE de su elección.
4. El trabajo deberá entregarse en formato digital (tanto las fuentes, ejecutables y documentación) el día y hora indicados. Esto en una carpeta comprimida bien identificada según plan de curso y compartido mediante el enlace en el AV.
5. El proyecto deberá ser creado utilizando control de versiones y el repositorio debe ser privado, solo compartido con el profesor y el compañero(s) de proyecto. Para este proyecto todo se debe desarrollar sobre la rama develop, y una vez finalizado el proyecto los estudiantes deberán fusionarlo con la rama main.
6. Parte del control de versiones al escribir el código fuente, es trabajar en pequeñas tareas y su commits respectivo, es importante que se evidencien todos los commits en el desarrollo, con mensajes claros de lo que se realizó en cada acción y quién lo hizo.
7. El proyecto debe ser desarrollado utilizando el modo gráfico con la biblioteca *CIMG*.

8. Cualquier trabajo práctico, que no sea de elaboración original de los estudiantes o haya sido copiado o adaptado de otro origen (plagio), de manera parcial o total, o se determina que existen dos o más trabajos con alguna similitud (por mínima que sea) se calificarán todos los involucrados con nota 0 (cero) y se procederá como lo indiquen los reglamentos vigentes de la universidad.
9. El proyecto deberá ser mostrado y defendido en la fecha indicada según cronograma. El porcentaje de la nota está sujeto a una demostración adecuada, cada participante debe defender el trabajo independientemente, y conocer la totalidad del código fuente, las ideas utilizadas y demostrar la autoría del código, el mostrar poco o nulo conocimiento puede influir en la nota y queda a criterio del profesor recibir el proyecto.
10. El proyecto estará conformado por 3 entregas en diferentes fechas:
    - Para la primera entrega del proyecto, consistirá en el UML y la explicación de los requerimientos, es decir, un análisis previo al desarrollo del código del proyecto.

    Ejemplo de análisis previo utilizando historias de usuario:

Como	<Tipo de usuario>
Quiero	<Objetivo (empieza en infinitivo, hace solo una cosa específica)>
Para	<Beneficios, justificación>
Criterio de aceptación	<Cómo sé yo que sí se cumplió>
Observaciones	<Notas extra a tener en cuenta para esta funcionalidad>

Como	Lector de blog
Quiero	Agregar comentarios a los posts y recibir alertas cuando me respondan.
Para	Poder agregar mi opinión a la lectura y conocer lo que opinan los demás lectores
Criterio de aceptación	¿Puedo agregar un comentario a un post específico? ¿Puedo modificar el comentario? ¿Recibo una alerta en mi correo cuando alguien responde cada uno de mis comentarios?
Observaciones	Se debe solicitar un correo válido para poder agregar comentarios y solo un usuario autenticado puede editar los comentarios.

Como	Jugador
Quiero	Acelerar el carro con el teclado
Para	Poder cambiar de velocidad en la partida y poder conducir mejor el carro
Criterio de aceptación	¿El carro se puede acelerar con una tecla específica? ¿El carro se puede desacelerar con una tecla específica?
Observaciones	Puede ser que el jugador requiere dos opciones de teclas para esta funcionalidad, por ejemplo, las **arrow keys** y utilizar el **AWSD**

Para este proyecto se esperan entre unas 15 a 25 historias de usuario. Evite repetir funcionalidad, pero abarque todo el proyecto.

El diagrama de clases será de manera general y único para observar las relaciones de clases, omita los Get y Set ya que se entiende que se utilizarán y solo agregan ruido al diagrama.

- Para la segunda entrega se espera la corrección de los cambios solicitados con respecto al avance inicial, la creación de las clases y funciones principales que se utilizarán en la aplicación.
- La entrega final estará conformada con las fuentes del proyecto, demostración en vivo y la defensa del código fuente por cada participante.

11. Fechas del proyecto:
    1. Entrega del enunciado: 22 de diciembre de 2022
    2. Primera entrega: Día 7 - 18 de enero de 2023
    3. Segunda entrega Día 10 - 26 de enero de 2023
    4. Entrega final: Día 13 - 09 de febrero de 2023

Antecedentes

¿Qué es CATAN? (CATAN, 2022)

El juego de mesa Los Colonos de Catán como se tradujo al español inicialmente y ahora solo Catán, es un juego de diseñado por Klaus Teuber, publicado por primera vez en Alemania en 1995 como Die Siedler von CATAN, luego en inglés como The Settlers of CATAN en 1996, CATAN celebró su 25 aniversario en el 2020. Este juego permite que un grupo de jugadores (de 3 a 4) puedan usar una variedad de estrategias para ganar puntos, por consiguiente, cada partida que se juega es diferente y única.

Ver video explicativo mostrado en el siguiente enlace: https://youtu.be/N5SljJbSRgc.

Según la temática del juego – las mujeres y los hombres de tu expedición construyen los dos primeros asentamientos en la isla de CATAN. Afortunadamente, la tierra aún deshabitada es rica en recursos naturales. Construyes caminos y nuevos asentamientos que eventualmente se convierten en ciudades. ¿Lograrás obtener la supremacía en Catán? El comercio de trueque domina la escena. Algunos recursos los tienes en abundancia, otros recursos son escasos. Mineral por lana, ladrillo por madera; comercia de acuerdo con lo que se necesita para sus proyectos de construcción actuales. ¡Proceda estratégicamente! Si encontró sus asentamientos en los lugares correctos e intercambió hábilmente sus recursos, entonces las probabilidades estarán a su favor. Pero tus oponentes también son inteligentes.

Figura 1. Juego de mesa Catán.

Para comenzar el juego, se debe construir el tablero de juego usando fichas de terreno hexagonales, es así como nace Catán, la cual consiste en una isla con montañas, pastos, colinas, campos y bosques, rodeada por el mar.

¿Cómo se juega? (Teuben, 2020)

Componentes

19 hexágonos de terreno

- 4 Bosques
- 4 Pastos
- 3 Sembrados
- 3 Cerros
- 3 Montañas
- 1 Desierto
- 6 Piezas de marco con 9 puertos

Figura 2. Fichas de terreno.

95 cartas de materias primas (19 de cada tipo)

- Madera = troncos = del Bosque
- Lana = oveja = del Pasto
- Cereal = gavilla de trigo = del Sembrado
- Arcilla = ladrillos = del Cerro
- Mineral = roca = de la Montaña

Figura 3. Cartas de materias primas.

25 cartas de desarrollo

- 14x Caballero
- 6x Progreso
- 5x Puntos de victoria

Figura 4. Cartas de desarrollo.

4 tablas de costes de construcción

Figura 5. Costes de construcción.

2 tarjetas especiales

"Mayor ruta comercial"

"Mayor ejército"

Figura 6. Tarjetas especiales.

Figuras de ciudades (4 colores)

- 16x Ciudades
- 20x Poblados
- 60x Carreteras

Montaje de la isla

Aunque hay diferentes formas de montar la isla, montaje para principiantes (distribución de fichas de terreno en un orden establecido) o montaje para jugadores avanzados, en donde el tablero de juego se acomoda aleatorio.

Preparación

1. Cada jugador elige un color y recibe 5 poblados, 4 ciudades y 15 carreteras, ni más ni menos. Cada jugador coloca 2 carreteras y 2 poblados sobre el tablero de juego, y deja los poblados, carreteras y ciudades restantes delante de él. Si sólo hay 3 jugadores, se retiran todas las piezas rojas del juego.
2. Cada jugador recibe una carta de "Costes de construcción".
3. Las cartas especiales **Mayor ruta comercial** y **Mayor ejército** se dejan junto al tablero de juego, al igual que los dos dados.
4. Las cartas de materia prima se separan por materia y se dejan boca arriba en 5 montones en los compartimentos del portacartas. El portacartas se deja junto al tablero de juego.
5. Las cartas de desarrollo se barajan y se dejan boca abajo en el compartimento libre del portacartas.
6. Por último, cada jugador recibe de cada poblado las primeras cartas de materia prima: por cada campo con el que limite el poblado, coge del montón una carta de la materia prima correspondiente.
7. Los jugadores deben mantener las cartas de materia prima ocultas en su mano.

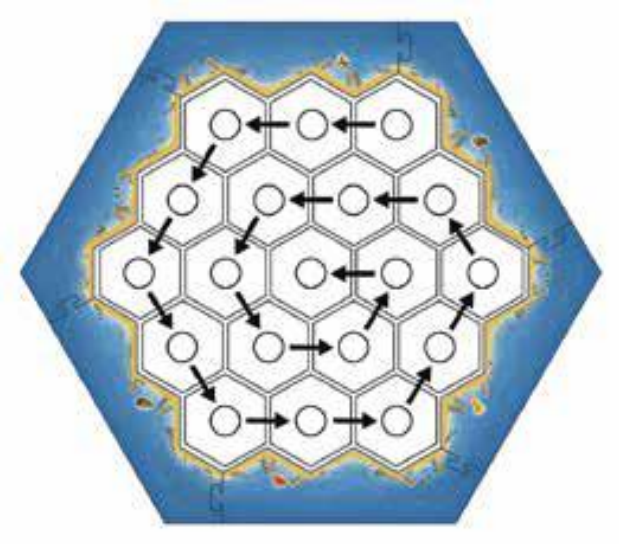


Figura 7. Colocación de las fichas de terreno y las fichas numeradas.

Desarrollo del juego (resumido)

El jugador de más edad comienza. En su turno, el jugador puede hacer varias cosas distintas, siempre siguiendo este orden:

1. Tirar los dados para determinar cuáles son las materias primas que se van a obtener en este turno. El resultado de los dados vale para todos los jugadores.
2. **Comerciar:** intercambiar materias primas, solo o con los otros jugadores.
3. **Construir:** carreteras, poblados o ciudades, y/o comprar cartas de desarrollo.

- **Además**, en cualquier momento de su turno, el jugador puede jugar una de sus cartas de desarrollo.

Después, el turno pasa al siguiente jugador por la izquierda, que comienza otra vez desde el punto 1.

Desarrollo del juego (detallado)

1. **Obtener materias primas:**
    a. El jugador comienza su turno tirando los dos dados: la suma de los resultados determina qué campos van a producir beneficios.
    b. Cada jugador que tenga un poblado en una encrucijada que limite con uno de estos campos obtiene una carta de materia prima de ese campo.
2. **Comercio:** Ahora, el jugador puede intentar comerciar (de cualquiera de las dos formas que se indican) para conseguir las cartas de materia prima que le faltan.
    a. Comercio interno: El jugador puede intercambiar cartas de materia prima con todos los demás jugadores. Puede decir qué materias primas necesita y cuáles está dispuesto a ofrecer a cambio. También puede escuchar las ofertas de los demás jugadores y hacer contraofertas. **Importante:** sólo se puede comerciar con el jugador al que le toca en ese momento. Los demás jugadores no pueden comerciar entre ellos.
    b. Comercio marítimo: El jugador, en su turno, también puede comerciar sin hacerlo con ningún otro jugador.
        i. En principio, siempre puede cambiar 4:1. Para hacerlo, devuelve cuatro cartas de materia prima iguales al montón y coge una carta de la materia prima que quiera.
        ii. Si ha construido un poblado junto a un puerto, puede comerciar de forma más ventajosa, a razón de 3:1, o incluso 2:1 en los puertos especiales (los que tienen un símbolo de materia prima).
        iii. **Importante:** el intercambio 4:1 siempre es posible, incluso cuando no se tienen poblados junto a los puertos.
3. **Construir:** Por último, en su turno el jugador puede construir y obtener así más puntos.
    - Para construir hay que pagar una determinada combinación de cartas de materia prima (según la carta de Costes de construcción); a cambio, el jugador coge de su reserva las carreteras, poblados o ciudades correspondientes y los pone sobre el tablero de juego.
    - Ningún jugador puede construir más edificios de los que posee, es decir, como máximo 5 poblados, 4 ciudades y 15 carreteras.

    a. **Carretera:** se necesita arcilla + madera.
        i. Una carretera nueva siempre tiene que partir de otra de tus propias carreteras, poblados o ciudades.
        ii. En cada camino sólo se puede construir 1 carretera.
        iii. En cuanto que un jugador construya una ruta que se componga de, por lo menos, 5 carreteras (las bifurcaciones no cuentan), obtendrá la carta especial la **Mayor ruta comercial**. Sin embargo, si otro jugador consigue construir otra ruta más larga que la del jugador que posee la carta, se quedará inmediatamente con dicha carta (y por tanto, con los 2 puntos).
    b. **Poblado:** se necesita arcilla + madera + lana + cereal

i. **Importante:** la "**regla de la distancia**": sólo se puede construir un poblado en una encrucijada, si las tres encrucijadas de alrededor NO están ocupadas por poblados (sean de quien sean).
ii. El poblado tiene que partir de una de las carreteras del jugador, obligatoriamente.
iii. Por cada poblado, su propietario obtiene materias primas de los campos limítrofes: 1 carta de materia prima de cada uno, cuando los dados así lo indiquen.
iv. Cada poblado vale 1 punto.

c. **Ciudad:** se necesita 3 minerales + 2 cereales.
i. Una nueva ciudad sólo se puede formar por la ampliación de un poblado.
ii. Si un jugador amplía uno de sus poblados a ciudad, retira la pieza del poblado y la pone en su reserva, sustituyéndola por una ciudad.
iii. Por cada ciudad, el jugador obtiene el doble de materias primas de los campos de alrededor: 2 cartas de materia prima cada vez, cuando los dados así lo indiquen.
iv. Cada ciudad vale 2 puntos.

d. **Comprar cartas de desarrollo:** se necesita mineral + lana + cereal.
i. Cuando un jugador compra una carta de desarrollo, coge siempre la primera del montón.
ii. Hay tres tipos diferentes de cartas de desarrollo, con distintos efectos en el juego: "Caballero", "Progreso" y "Puntos".
iii. Lo mejor es mantener las cartas compradas boca abajo hasta que se utilicen, para que ningún jugador pueda sacar conclusiones.

4. Excepciones

a. Si sale el siete, el ladrón entra en escena.
i. Si a un jugador le sale un 7, nadie recibe materias primas ese turno.
ii. Todos los jugadores que tengan más de 7 cartas de materia prima deberán deshacerse de la mitad de ellas (las que ellos quieran), dejándolas en sus montones correspondientes. En los números pares se redondea hacia abajo (p.ej., si un jugador tiene 9 cartas, tiene que deshacerse de 4).
iii. Después, el jugador está obligado a cambiar de sitio al ladrón, según se indica a continuación:
1. El jugador tiene que colocar inmediatamente al ladrón sobre la ficha numérica de otro campo, a su elección.
2. Después le roba a otro jugador que tenga una ciudad o poblado alrededor de ese campo, una carta de materia prima, escogiendo entre las cartas que el jugador "robado" mantiene boca abajo (sólo las de materia prima, no las de desarrollo). Si hay más de un jugador que tenga poblados o ciudades alrededor de ese campo, podrá elegir a qué jugador desea robarle la carta.
3. **Importante:** si sale el número sobre el que se encuentra el ladrón, los jugadores que tengan poblados o ciudades en esos números no reciben ninguna materia prima, y el ladrón se mantiene en el mismo lugar.

b. Jugar cartas de desarrollo.

i. El jugador puede decidir jugar (mostrar) una carta de desarrollo en cualquier momento de su turno, incluso antes de tirar los dados, siempre que NO sea una carta que haya comprado en ese mismo turno.
ii. Cartas de "Caballero":
    1. Quien juegue una carta de caballero, tiene que cambiar de sitio inmediatamente al ladrón, siguiendo los puntos 1 y 2 anteriores (salida en escena del ladrón).
    2. Las cartas de caballero que ya hayan sido jugadas permanecen boca arriba delante de su propietario.
    3. El jugador que primero tenga tres cartas de caballero boca arriba ante sí, obtiene la carta especial **Mayor ejército**, que vale 2 puntos.
    4. Cuando otro jugador tenga más cartas de caballero descubiertas ante sí que el poseedor actual del **Mayor ejército**, obtiene de inmediato la carta, y con ella, los 2 puntos.
iii. Cartas de progreso:
    1. Quien juegue una de estas cartas, deberá seguir las indicaciones contenidas en ella. Después, la carta se retira del juego (se deja otra vez en la caja).

c. Puntos:
    i. Las cartas con puntos se mantendrán boca abajo. No se pueden descubrir hasta que el jugador esté seguro de que tiene 10 puntos, y por lo tanto, de que va a ganar el juego (entonces se pueden descubrir todas a la vez).

5. Final del juego

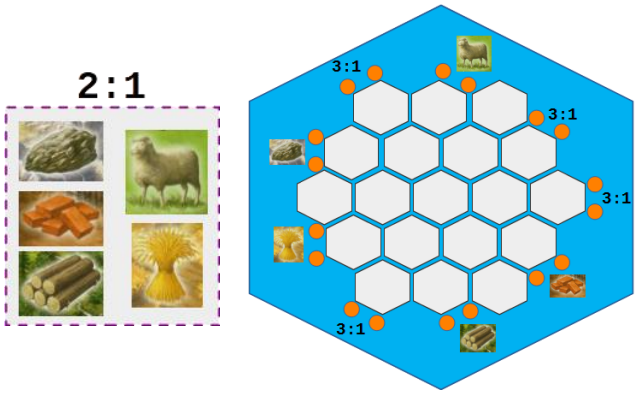
a. El juego acaba en el turno en el que un jugador consiga 10 o más puntos.
b. Para ganar, el jugador tiene que conseguir los 10 puntos en su turno.

Trabajo para realizar por los estudiantes

Basado en el juego Catán, el proyecto consistirá en hacer una versión en donde participan 3 o 4 jugadores, 2 humanos y 1 o 2 computadoras (3 humanos y una computadora).

1. Debido a que todas las ilustraciones del juego de mesa son propiedad del autor, queda prohibida su reproducción sin el expreso consentimiento del artista, por lo tanto, se deberá realizar cada una de las ilustraciones requeridas para el juego. Es decir, deberá utilizar su creatividad para desarrollar las ilustraciones de cada una de las cartas que permitan cumplir satisfactoriamente con cada una de las funcionalidades del juego.
2. Se requiere que el juego sea en modo gráfico, el diseño es libre, pero debe cumplir con mostrar la isla y al menos el retablo del jugador en turno. Deben poder identificarse todas las cartas visibles en pantalla y su contenido debe ser fácilmente legible.

3. Al inicio de la partida se debe realizar el equivalente a **barajar** las fichas de terreno. Es decir, debe ordenarse aleatoriamente, de modo que la secuencia de fichas sea diferente para cada partida, asimismo, todas las cartas que involucran el juego.
4. Se deberán cumplir con todas las reglas de las acciones del jugador en su turno según el juego original.
5. El progreso del juego deberá guardarse en un archivo y también poder cargarse en una posterior ejecución del programa para poder continuar la partida. Dentro del archivo, se deberá guardar la secuencia de las condiciones del juego, esto para que cuando se reanude la partida a partir del archivo se mantengan las mismas condiciones del juego.
6. Se deben utilizar todas las estructuras de datos (arreglos, listas enlazadas, grafos, etc.) necesarias que permitan almacenar las cartas, el mapa, los poblados, las ciudades, los caminos en memoria correspondientes, y los retablos de los jugadores. (se sugiere la utilización de herencia de clases y objetos polimórficos).
7. Deben especificarse en pantalla los controles de teclado y/o ratón en los momentos oportunos o durante todo el tiempo, de modo que sea fácil interactuar con el programa. Debe incluir controles necesarios para jugar (ej. tomar, ubicar, terminar turno, etc.) y también los de Iniciar/Guardar/Cargar partidas y Salir del juego.
8. Es importante que en todo momento los jugadores conozcan el estado actual del juego, quién debe jugar, qué opciones hay y demás aspectos relacionados.
9. Al final de la partida el programa debe hacer los conteos de puntos de cada jugador e indicar el ganador.
10. Los Puertos para el comercio de los recursos siempre se mantendrán en la misma posición para todos los juegos que se creen, según el siguiente diagrama:



Referencias bibliográficas

- CATAN. (2022). Recuperado de https://www.catan.com/catan
- Teuber, Klaus. (2020). *CATAN Game Rules & Almanac*. Recuperado de https://www.catan.com/sites/default/files/2021-06/catan_base_rules_2020_200707.pdf

Criterios de evaluación

1. Eficacia (Cumple el funcionamiento solicitado) (65%).

Rubro	Deficiente	Regular	Satisfactorio
1. General (15%)			
1.1. El programa permite definir los parámetros iniciales del juego (3%)	**0**	**1**	**3**
1.2. El programa cuenta con controles para iniciar, jugar, guardar, cargar y salir del programa (2%)	**0**	**1**	**2**
1.4. El programa guarda en un archivo la jugada y permite recuperarla para continuar (10%)	**0**	**5**	**10**
1.5. Cada estudiante atiende las preguntas y dudas presentadas por el profesor en esta sección, muestra el código y entiende cada parte de este. Nota individual.	**-15**	**-8**	**0**
2. Graficado (15%)			
2.1. El programa permite que el usuario pueda utilizar el ratón (mouse) para controlar el entorno del programa (5%)	**0**	**3**	**5**
2.2. El diseño de la interfaz es acorde y amigable con el usuario y facilita el manejo de las funcionalidades del programa (10%)	**0**	**5**	**10**
3. Durante una partida (35%)			
3.1. Al ejecutar el juego todas las jugadas cumplen con las reglas indicadas (10%)	**0**	**10**	**10**
3.2. El programa calcula los puntos correctamente al finalizar el juego y determina quién gana la partida (5%)	**0**	**3**	**5**
3.3. Se utilizan las estructuras de datos necesarios para manejar la información (20%)	**0**	**5**	**15**
3.3. Cada estudiante atiende las preguntas y dudas presentadas por el profesor en esta sección, muestra el código y entiende cada parte de este. Nota individual.	**-35**	**-16**	**0**

Satisfactorio: cumple todo lo solicitado o por lo menos un 90%

Regular: cumple la mayoría de la funcionalidad solicitada entre un 70% a un 90%, presenta errores menores en la lógica, y requiere de cambios menores para completar la aplicación.

Deficiente: No presenta la funcionalidad, es muy escasa, no se puede utilizar la aplicación, está confusa, no se entiende ni por el estudiante o profesor, o es diferente a lo solicitado.

2. Eficiencia y estructura solicitada (15%)

Rubro	Deficiente	Regular	Satisfactorio
Separación lógica de capas (5%)	Se desarrolló la app en una sola clase o no se utilizó clases **(0%)**	Se utilizan clases con funciones y atributos como contenedores de información, pero no hay división clara de lógicas. **(2%)**	Los nombres de clases, atributos y métodos correspondientes realizan lo que su nombre sugiere. **(5%)**
Tratamiento de errores y validación (5%)	El programa se detiene continuamente porque no tiene control de errores o este es mínimo. **(0%)**	El programa tiene control de muchos errores a nivel de código, pero se ve interrumpido debido a la manipulación del usuario. Presenta errores de lógica que desembocan en comportamientos inesperados. **(2%)**	El programa tiene el control interno de todos los errores predecibles, e informa al usuario de los errores que surjan por la manipulación del programa. **(5%)**
Ejecución (5%)	El programa no compila o compila, pero no hace nada. **(0%)**	El programa corre, pero no realiza todas las funciones solicitadas. **(2%)**	El programa corre y realiza todas las funciones solicitadas. **(5%)**

3. Control de versiones (5%)

Rubro	Deficiente	Regular	Satisfactorio
Utiliza control de versiones y cuentas separadas (1%)	No utilizan control de versiones. **(0%)**	Utilizan control de versiones, pero todo desde una misma cuenta. **(0.5%)**	Cada integrante utiliza una cuenta y el repositorio es privado, lo comparten con el profesor. **(1%)**
Utilizan ramas, el archivo readme.md y los comandos merge correctamente (1%)	Todo el proyecto está hecho en el main, no presentan o actualizan el archivo readme.md, tienen problemas con el merge del proyecto o no pueden mover el código a la rama main. **(0%)**	Presentan la rama develop y main, el archivo readme está utilizado, pero no actualizado. Tienen problemas con el merge y presentan el código en el develop. **(0.5%)**	Utilizan correctamente las ramas, develop para trabajar, main para la versión final y la integración de códigos. El readme está actualizado con las instrucciones para correr el juego en desarrollo e indica cómo organizaron el código. **(1%)**
Utiliza correctamente los commits (3%)	Hay pocos commits, no tienen comentarios en inglés o no indican que hacen, son commits reactivos, involucran múltiples tareas. **(0%)**	Utilizan commits, pero algunos son reactivos, no tienen un formato comprensible de lo que hacen. **(1%)**	Los commits corresponden a tareas premeditadas y hacen una sola cosa, están bien comentados, y están en idioma inglés. **(3%)**
Defensa del proyecto, se evalúa individualmente (hasta -5% de este rubro)	Hay un desbalance en la codificación que se evidencia en los commits, la mayoría del código o lo más importante lo hace un solo estudiante. El estudiante no puede indicar para que se hace un commit específico o defender el código de este. **(-5%)**	Los commits están balanceados pero el estudiante no puede indicar para que se hace un commit específico o defender el código de este en su totalidad. **(-2%)**	Los commits tienen sentido, siguen un orden lógico, los realizan de forma equitativa, los pueden defender cada participante. **(0%)**

4. Creatividad/Originalidad (15%)

Rubro	Deficiente	Regular	Satisfactorio
Documentación (4%)	No presenta documentación en la fecha indicada o no la corrige. **(0%)**	La documentación no está completa, no contempla toda la información relevante para resolver el problema. **(2%)**	La documentación es abundante, completa y bien formada. Presenta originalidad y creatividad en las ideas utilizadas para resolver el proyecto. **(4%)**
Diagramación UML (4%)	No presentó diagramación UML de las clases o los diagramas no corresponden a los contenidos teóricos vistos en clase, solo muestran una estructura que no se sigue en la implementación. **(0%)**	Los diagramas están incompletos, no presentan elementos importantes a considerar o no se actualizan con la versión final del proyecto. **(2%)**	Los diagramas UML corresponden a los conceptos teóricos vistos y hay correspondencia entre lo modelado y lo implementado en código tanto en la parte de definición de clases como en su respectiva implementación. **(4%)**
Creatividad y Originalidad (2%)	En el proyecto no se demuestra innovación, usa imágenes descargadas de internet o bibliotecas no vistas durante el curso. **(0%)**	Diseñó todo, pero de forma muy básica y poco intuitiva. **(1%)**	Demuestra capacidad de generar nuevas ideas o conceptos que produjeron soluciones originales. Utiliza un diseño coherente y trabajado. **(2%)**
Fuentes (5%)	El código fuente es desordenado y no especifica lo que hace, no sigue las buenas prácticas. **(0%)**	Existen algunas pautas de buenas prácticas utilizadas, pero no es coherente en todo el código, requiere separar lógica en muchas funciones o refactorizar, los nombres no son claros, entre otros problemas comunes. **(2%)**	Utiliza las buenas prácticas de desarrollo en C++ vistas en clase, es coherente con el estilo en todos los archivos. el código es limpio. **(5%)**