99.7% Citric Liquid

Proyecto semestral

Profesores: Matías Toro y Nancy Hitschfeld
Auxiliares: Daniel Ramírez, Ignacio Slater, Joel Riquelme y Vicente González
Semestre: Primavera 2023

1. Contexto

En el presente proyecto, usted diseñará una versión simplificada del juego *100% Orange Juice*, una obra original de *Orange Juice* y distribuido por *Fruitbat Factory*. Se trata de un juego de tablero en el que participan cuatro jugadores. El reto principal es alcanzar ciertos requisitos y ascender al nivel más alto, superando a los demás jugadores en la competencia.

A continuación se explicarán los aspectos a implementar del videojuego. Es importante leer el documento completo para entender cada aspecto particular, pues muchos están definidos en base a otros, o se componen parcial o completamente, directa o indirectamente de ellos.

2. Descripción

2.1. Unidades

Se define como unidad a aquella entidad que participará en el juego, ya sea como un personaje, controlable por el usuario, o como un *Wild Unit*, no controlable por el usuario. A partir de este momento se usarán los términos usuario y jugador de manera indistintiva, y puede ser utilizada para hacer referencia al personaje que es controlado por el usuario, mientras que personaje hará referencia exclusivamente a la entidad controlable por dicho usuario.

Todas las unidades tienen una cantidad de *Hit Points* máximos y una cantidad de *Hit Points* actuales. También, poseerán una determinada cantidad de puntos de daño (*ATK*), defensa (*DEF*) para protegerse ante ataques y evasión (*EVA*) para intentar esquivar un ataque por completo.

Cuando los *Hit Points* actuales de un jugador llegan a 0, entonces el personaje quedará en un estado de *K.O*, mientras que los *Wild Unit* simplemente desaparecerán del panel en el que se encontraban.

2.1.1. Personaje

Mencionado anteriormente, un personaje es una entidad controlable por el jugador. Estos personajes tendrán la capacidad de interactuar entre ellos y otros tipos de unidades.

Cada personaje cuenta con un número entero de estrellas que puede aumentar o disminuir en el desarrollo de la partida. Al comienzo del turno de un jugador, este ganará una cantidad de estrellas igual a $\lfloor Chapters \ / \ 5 \rfloor + 1$.

Los personajes también cuentan con un contador de victorias. Las victorias se obtienen al ganar combates, y la cantidad de victorias que se gana al combatir dependerá del tipo de enemigo enfrentado. La cantidad de puntos de victoria por cada tipo de enfrentamiento son:

- *Wild Unit*: 1 Victoria

- Otro jugador: 2 Victorias

Cuando un personaje es derrotado y pasa a estado de *K.O*, entrará a una fase llamada *Recovery*, donde el jugador **no puede** jugar su turno. Para salir de esta fase, el jugador debe lanzar un dado y obtener una cantidad mayor o igual a la requerida para recuperarse. Si logra recuperarse, entonces puede jugar su turno inmediatamente.

La cantidad necesaria para recuperarse comienza en 6 y va disminuyendo en 1 a medida que pasan los *Chapters*.

2.1.2. *Wild Unit*

Los *Wild Units* son enemigos de los personajes. Estos se generarán de manera aleatoria en los *Encounter Panel*, y se mantendrán en ese panel hasta que sean derrotados. Si derrotan a un personaje, este perderá la mitad de sus estrellas, redondeado hacia abajo, y serán transferidas hacia el *Wild Unit*, que a su vez entregará la cantidad de estrellas que tenga en el momento que sea derrotado.

Estos son los bellacos disponibles que lucharán contra los jugadores:

Enemigo	HP	ATK	DEF	EVA
Chicken	3	-1	-1	+1
Robo ball	3	-1	+1	-1
Seagull	3	+1	-1	-1

Tabla 1: Wild Units

2.2. Norma

Para ganar una partida, un jugador debe alcanzar un *Norma 6*, antes que el resto de los participantes. Todos los jugadores comienzan la partida con *Norma 1* y deben cumplir una serie de objetivos específicos para subir su nivel de *Norma*.

El jugador puede elegir uno de entre dos objetivos para subir su nivel de *Norma*, estos son: alcanzar una cierta cantidad de estrellas o conseguir cierta cantidad de victorias. El objetivo se debe elegir una vez que dicho jugador sube de nivel y no puede ser cambiado hasta que sea completado.

Cuando se cumple el requisito para ganar una *Norma*, El jugador debe pasar por un *Home Panel* para activar el efecto de *Norma Check*, y así realizar la subida de nivel. A esto se le conoce como *Norma Clear*.

Norma	Estrellas	Victorias
2	10	1
3	30	3
4	70	6
5	120	10
6	200	14

Tabla 2: Objetivos para subir de nivel

2.3. Tablero

El tablero es el lugar donde se juega la partida y se ubican los jugadores. Está formado por paneles conectados entre sí donde:

- Cada panel tiene un tipo,
- Puede estar siendo ocupado por uno o más jugadores, y
- Tiene uno o más paneles siguientes.

Al inicio de su turno, el jugador lanzará un dado de 6 caras, y se moverá esa cantidad de paneles, en la dirección que indiquen. En el caso que se pase por un panel que tenga más de un panel siguiente, entonces el jugador debe elegir cuál camino tomar.

Existen múltiples tipos de paneles, que se especificarán a continuación.

- ***Neutral Panel***: Este tipo de panel no tiene ningún tipo de efecto sobre el jugador, si cae aquí entonces terminará su turno sin novedades.
- ***Home Panel***: Cada jugador tiene un panel de hogar personal que se ubica cuando comienza el juego, estos se activan de dos maneras distintas:
  - Si el jugador es el dueño del panel, puede elegir detenerse en él al pasar sobre este incluso si le quedan movimientos disponibles.
  - Si el jugador no es el dueño del panel, entonces solo se activará si cae exactamente en este.

  En ambos casos, el turno terminará luego de activar el panel. Al activarse, el jugador recuperará un punto de vida y el panel realizará un *Norma Check*.
- ***Bonus Panel***: Este panel otorga estrellas al jugador. Cuando cae en uno, el jugador debe lanzar un dado de 6 caras y ganará una cantidad de estrellas igual a $min(roll \cdot Norma, roll \cdot 3)$.
- ***Drop Panel***: Este panel quitará estrellas al jugador. Cuando cae en uno, el jugador debe lanzar un dado de 6 caras y perderá una cantidad de estrellas igual a $roll \cdot Norma$.
- ***Encounter Panel***: Al caer en este panel, el jugador entrará en combate con un *Wild Unit* aleatorio.

2.4. Combates

Los combates son eventos que pueden darse entre dos jugadores o entre un jugador y un *Wild Unit*. Las batallas son desencadenadas cuando un personaje cae en un panel específico. Se realizan de la siguiente forma:

1. El jugador que activó el combate lanza el dado para decidir su cantidad total de ataque.
2. Se le suma la cantidad obtenida al ataque base del jugador.
3. La unidad aludida debe elegir entre defender o esquivar, y luego lanzar un dado:
   - Si se escoge la opción **defender**, entonces la unidad defensora recibirá un daño equivalente a $max\big(1, roll_{atk} + ATK - \big(roll_{def} + DEF\big)\big)$

- Si se escoge **esquivar**, si $roll_{eva} + EVA > roll_{atk} + ATK$ este no recibirá daño, y en caso contrario recibirá el daño total del ataque.

4. Si la unidad que recibió el ataque le quedan *Hit Points* actuales, entonces devolverá un ataque de la misma manera, y luego finalizará el combate.

2.4.1. Jugador contra Jugador

Estos combates se pueden activar cuando un jugador cae un panel en el que hayan uno o más personajes. Si decide enfrentarse a algún jugador, se iniciará un combate entre ambos.

Si alguno de los jugadores reduce los *Hit Points* de su oponente a 0, entonces aumenta su contador de victorias en 2 y se le transfieren la mitad, aproximada hacia abajo, de las estrellas del oponente.

2.4.2. Jugador contra *Wild Unit*

Este tipo de combate se inicia cuando un jugador cae en un *Encounter Panel*. En estos combates el jugador siempre atacará primero y el enemigo elegirá al azar entre defender o esquivar. Si el enemigo derrota al jugador, entonces el jugador perderá la mitad de sus estrellas y el enemigo se quedará con estas. Cuando un jugador derrota a un enemigo, todas sus estrellas pasarán al jugador, y este aumentará su contador de victorias en 1.

Cada vez que se desencadena por primera vez un combate contra un *Wild Unit*, se elige al azar entre uno de los 3 bellacos existentes.

Luego de los combates, los enemigos mantienen sus *Hit Points* actuales con los que quedaron al final de estos. Si son derrotados, entonces se creará uno nuevo, con su salud completa y sin estrellas.

2.5. Sistema de turnos

Cada partida del juego se divide en *Chapters*, que a su vez se dividen en 4 turnos cada uno, uno por cada jugador. Al comienzo de la partida se asigna el orden de los turnos a los jugadores de manera aleatoria y se mantiene ese mismo orden a lo largo de toda la partida. Cada vez que los 4 jugadores terminan sus respectivos turnos, entonces se pasa al siguiente *Chapters*, que en este caso es solamente un contador.

El turno de cada jugador se estructura de la siguiente manera:

1. Si está en estado de *K.O* entonces pasa a la fase de *Recovery*.
2. Recibe la cantidad de estrellas especificadas en la sección 2.1.1
3. Lanza un dado, y se mueve en la cantidad de paneles indicadas por el resultado, exceptuando los casos en los que puede detenerse anticipadamente.
4. Se activa el efecto del panel.
5. Se pasa al turno del siguiente jugador.

3. Modelo de la solución

La resolución de este proyecto se hará siguiendo el patrón arquitectónico *Modelo-Vista-Controlador*, donde primero se implementará el *Modelo*, luego el *Controlador* y finalmente la *Vista*. Este patrón se irá expli-

cando a lo largo del proyecto, y tendrán una guía constante para su correcta resolución. En el contexto del proyecto, estos componentes serán como se explica a continuación:

Modelo – Para la primera parte se le solicitará que cree todas las entidades necesarias que servirán de estructura base del proyecto y las interacciones posibles entre dichas entidades. Las entidades en este caso se refieren a los elementos que componen el juego.

Vista – Se le pedirá, de manera opcional, que cree una interfaz gráfica simple para el juego que pueda responder al input de un usuario y mostrar toda la información relevante del juego en pantalla. Esta interfaz puede ser simplemente en la consola.

Controlador – Servirá de conexión lógica entre la vista y el modelo, se espera que el controlador pueda ejecutar todas las operaciones que un jugador podría querer efectuar, que entregue los mensajes necesarios a cada objeto del modelo y que guarde información importante del estado del juego en cada momento.

4. Evaluación

4.1. Puntaje

- **Código fuente (4.0 pts.):** Este ítem se divide en dos partes:
  - **Funcionalidad (1.5 pts.):** Se evaluará que su código provea la funcionalidad solicitada. Para esto, se exigirá que testee las funcionalidades que implementó. **Si una funcionalidad no se testea, no se podrá comprobar que funciona, por lo tanto, no se asignará puntaje en este apartado por ella.**
  - **Diseño (2.5 pts.):** Se evaluará la calidad de su código, exigiendo que este cumpla con los principios de diseño enseñados en el curso.
- **Coverage (1 pt.):** Se evaluará que su código tenga un coverage de al menos 50% de las **líneas** de código. Para obtener el puntaje completo, el coverage debe ser de al menos 90%.
- **Documentación (1 pt.):** Cada clase, interfaz y método público debe estar documentado. Siga la guía de documentación que encontrará en el *FAQ* del curso.

Para guiar la realización de este proyecto, se plantearán objetivos pequeños en forma de *entregas parciales*.

- **Entrega Parcial (0.5 pts. c/u):** Dichas entregas deberán ser enviadas en la fecha señalada y no serán evaluadas directamente con nota. Solamente será evaluado que al momento de entregarla se cumpla con el objetivo planteado, **independientemente de su diseño**.

La no entrega de una entrega parcial supondrá un descuento en su nota final, en la cantidad de puntos señalada. Su finalidad es solamente incentivar el trabajo constante y continuo, por sobre realizar las tareas a último minuto, por lo que las entregas parciales **no tendrán feedback**.

4.2. Modalidad de entrega

Deberá subir todo su trabajo al repositorio privado proporcionado a través de *GitHub Classroom*.

La entregas consistirán en un archivo de texto (`.txt`) que se subirá en la sección *Tareas* de *U-Cursos*. Dicho archivo deberá tener el siguiente formato:

```
Nombre: <Nombre completo>
PR: <URL del pull request>
```

Cuando considere que su entrega está lista para ser evaluada, realice un ***pull request*** a la rama *main* de su repositorio. Un «pull request» es básicamente proponer cambios y solicitar que alguien revise y aplique esos cambios en GitHub. Si después realiza *push* de más commits a esa misma rama, el pull request se actualizará automáticamente y se considerará la versión más reciente para revisión.

Si necesita seguir trabajando en su tarea después de haber creado el pull request, simplemente cree una nueva rama y continue su trabajo desde ahí.

4.3. Bonificaciones

Puede ganar puntos adicionales al implementar lo siguiente:

- **TAREA 1,2,3 – Readme (0.2 pts.):** Obtenga puntos adicionales detallando en su `README.md` cosas como:
  - ***¿Qué funcionalidades implementó?***
  - ***¿Por qué tomó ciertas decisiones en su diseño?***
  - ***¿Cómo está organizado su código?***
  - ***¿Qué patrones de diseño utilizó?***
- **TAREA 2,3 – Manejo de excepciones (0.3 pts.):** Se valorará el manejo adecuado de excepciones. ***Bajo ninguna circunstancia debe una excepción ser expuesta al usuario*** .
- **TAREA 3 – Interfaz gráfica (0.5 pts.):** Si incorpora una interfaz gráfica (puede ser de consola) para el juego, se podrá optar a esta bonificación.
- **TAREA 3 – Patrones de diseño adicionales (0.5 pts.):** Al incorporar correctamente más patrones de diseño de los solicitados, se podrá optar a esta bonificación.

5. Recomendaciones

- **La funcionalidad no es suficiente:** Priorice la calidad del diseño y aplique las buenas metodologías de programación aprendidas. No se aferre al primer diseño; busque oportunidades de mejora y extensibilidad. Pregúntese, ¿Qué pasaría si en el futuro se quisiera implementar X funcionalidad nueva? ¿Mi código tendría que ser modificado si quisiera hacerlo? De esta manera, la mayor parte de su código soportaría extensibilidad para cualquier funcionalidad nueva que se desee agregar.
- **NO DEJE LA TAREA PARA ÚLTIMO MOMENTO**. Aunque es un consejo común, y es algo que se dice en todos los cursos, aquí es particularmente muy importante. Trabajar apresuradamente le resultará en un diseño descuidado, no tendrá tiempo de pensar en uno mejor, y no aplicará el contenido enseñado en el curso.
- Aunque no es obligatorio, considere seriamente documentar su programa en inglés. La mayoría de la documentación en programas *open-source* está en este idioma, utilice esta oportunidad para practicar su inglés.

- **Si algo no se especifica en el enunciado, tiene libertad creativa**, esto es, manejar casos de borde o ciertos aspectos funcionales que no se detallan, pero asegúrese de cumplir con lo solicitado y aplicar los contenidos aprendidos del curso.
- Puede utilizar cualquier funcionalidad que el lenguaje es capaz de proveerle, independiente de si se enseñó en el curso o no, con el fin de resolver algún apartado de su proyecto. Sin embargo, asegúrese que lo está utilizando de la manera correcta.
- **PREGUNTE, CONSULTE Y AYUDE**. Este es un curso donde las dudas sobre el proyecto, la materia y los diseños abundan. No tenga miedo de enviar un mensaje al foro, seguramente haya más de una persona con el mismo problema, y también alguien que ya lo haya resuelto. Apóyese en el equipo docente y sus propios pares, podrá avanzar de manera mucho más expedita.