



Universidad de la Habana
Facultad de Matemática y Computación
Licenciatura en Ciencias de la Computación

The Clam Boat a Trading Cards Games

Alejandro Lamelas Delgado

@Artagos

Alejandro Ramirez Trueba

@AleTheCreation

Mauro Eduardo Campver Barrios

@Mauro - 02



Índice

1. Introducción	2
2. Estructura	4
2.1. Game	4
2.2. Interpreter	6
2.3. Visual	8



1. Introducción

Gwent, un juego de cartas colecciónables desarrollado por CD Projekt RED, nos inspiró para crear nuestro propio juego de cartas colecciónables. La mecánica del juego es simple: creas una baraja de 30 cartas y te enfrentas a tu oponente en tres rondas. El objetivo es burlar a tu oponente con creatividad y estrategia. Con Gwent, vimos cuánto potencial hay para explorar dentro del género de los juegos de cartas colecciónables. Como tal, utilizamos nuestra experiencia con los juegos de mesa, así como nuestro interés en los juegos de estrategia, para diseñar mecánicas en torno a la gestión de recursos, la construcción de mazos y el pensamiento rápido que ayudaría a los jugadores a formar sus propias estrategias para la victoria. A través del ejemplo de Gwent, nos inspiramos para crear un nuevo tipo de juego de cartas colecciónables que no esté atado por reglas rígidas o cartas dominadas. Esto nos permitió desarrollar un meta en constante evolución donde los jugadores pueden crear mazos que son únicos y poderosos a su manera. Al ver lo que Gwent era capaz de hacer con su giro único en la fórmula tradicional del juego de cartas colecciónables, nos sentimos capacitados para hacer lo mismo y crear algo nuevo.



Figura 1: Menú de inicio



Nuestro juego se desarrolla en un mundo dividido en cuatro facciones:

1. La Facción Neutral estaba compuesta por todos los seres vivos que deseaban permanecer independientes y no alineados. Su fuerza estaba en su neutralidad y capacidad para trabajar con otros, independientemente de su facción. Al vivir alienados del mundo, desarrollaron una especie de resistencia a la magia, por lo que vencen con facilidad a la facción Scoia'tael, aunque se hicieron débiles a las artes oscuras de Nilfgaard
2. La Facción Scoia'tael era un grupo resistente de elfos, enanos y humanos dedicados a proteger la naturaleza de los monstruos y la malvada facción Nilfgaard. Intentaron mantener el equilibrio entre la naturaleza y la civilización a través de su destreza militar y magia druídica. Sus habilidades mágicas le daban ventaja sobre los Monster, aunque sufrían en batalla ante la facción Neutral
3. La Facción de Monster estaba compuesta por criaturas previamente temidas por todas las demás razas: dragones, gigantes, trolls y similares. Monstruosos o no, estos seres lucharon por sus propios intereses, ya sea por renombre, territorio u oro, pero sin embargo se ganaron el respeto como enemigos formidables. Sus presas favoritas son los seres de Nilfgaard y sus peores enemigos los Scoia'tael
4. La facción de Nilfgaard, un imperio malvado fundado por la oscura profecía del infame hechicero Vilgrivek que prometía un poder y un dominio aún mayores si su propio pueblo lo tomaba. Estos habitantes de la oscuridad tramaban sin cesar para adquirir más poder a través de cualquier medio necesario: guerras, diplomacia o explotación de recursos. Los altos estudios en artes oscuras los hacen implacables ante los seres Neutrales aunque caían con facilidad ante los Monster

Luego, estas cuatro facciones se enfrentaron entre sí en una lucha interminable por la supremacía. Cada facción aportó sus propias fortalezas y debilidades mientras competían por una posición en esta pelea, elaborando hechizos y tácticas nunca antes vistas! Tu destino es libre, puedes unirte a una facción y luchar por la victoria de esta, o unirte a un grupo de rebeldes conformados por seres de todas las facciones, tú decides



2. Estructura

Nuestro trabajo se encuentra formado por dos proyectos, Una biblioteca de clases, implementada en C# 10, .NET Core 6, donde se implementa toda la funcionalidad lógica y una aplicación visual, implementada en Windows Form.

La biblioteca de clases se divide en dos carpetas *Game* e *Interpreter*:

2.1. Game

Dentro de Game encontramos las siguientes clases

1. Bot: Proporciona al programa la capacidad de hacer el juego automatizado, permite que el programa tome decisiones sobre cómo jugar cartas, qué cartas elegir y cuándo terminar el turno. La clase de bot puede evaluar los estados del tablero, considerar estrategias basadas en mazos o estrategias de oponentes y producir resultados procesables que el programa puede usar.
2. CardDataBase: Se utiliza para almacenar y administrar las cartas del juego, que se almacenaran en `public static List CardList = new List();`. Almacena información sobre cada tarjeta, como su nombre, descripción, facción y en caso de tenerlo, los efectos.

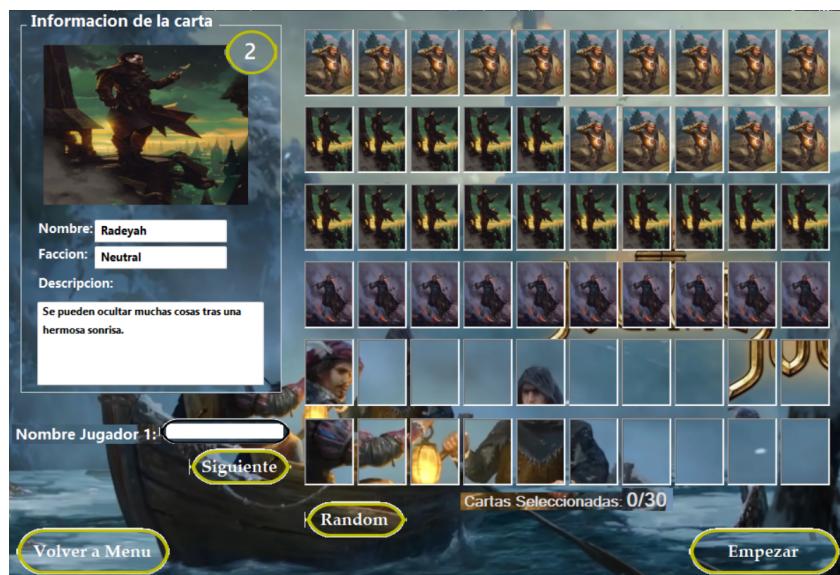


Figura 2: *Conformando deck*



3. Cards: En esta, se almacenan las propiedades relacionadas con las cartas individuales:

Listing 1: Card.cs

```
public class Card
{
    public int Id;
    public string Name;
    public string Description;
    public int BasePower;
    public int Power;
    public int Faction;
    public PictureBox image;
    public bool Passive;
    public List<efecto> Efectos;
}
```

4. GameRun: Es la responsable de administrar todo el juego de principio a fin. Contiene toda la lógica y los métodos necesarios para crear, ejecutar y administrar el juego. Las principales responsabilidades incluyen inicializar los mazos para ambos jugadores, manejar eventos basados en turnos, como robar cartas y jugarlas, manejar el combate, verificar las condiciones del final de cada ronda o del juego, como mazos vacíos, diferencia de power en el campo, turnos pasados y así, declarar un ganador, o terminar el juego en empate.
5. IPlayer: Interface encargada de las propiedades de los player

Listing 2: IPlayer.cs

```
public abstract class IPlayer
{
    public string Name { get; set; }
    public int RaundsWon { get; set; }
    public string TotalRounds { get; set; }
    public List<Card> PlayerM { get; set; }
    public List<Card> Graveyard { get; set; }
    public List<Card> Hand { get; set; }
    public List<Card> Deck { get; set; }
    public int TotalPoint { get; set; }
    public bool PassRound { get; set; }
```



```
public bool PlayerIsaBot { get; set; } }
```

6. Player: Se usa para representar a cada participante individual en el juego. Almacena información importante, como el deck, las cartas en mano, nombre, rondas ganadas y cartas en el tablero. Tiene métodos para robar cartas de su deck y actualizar la cantidad de power en juego

2.2. Interpreter

Dentro de Interpreter encontramos las siguientes clases

1. CardEffects: Permite la implementación de efectos aplicables a todas las cartas en el campo, tanto aliadas como enemigas, las cartas pueden pasar de no tener ningún efecto a tener una gran variedad de efectos relacionados entre sí
2. Condiciones: Se encarga de enlazar los efectos y establecer condiciones para la realización de estos

Listing 3: Condiciones.cs

```
{  
    public delegate bool Comprobacion (Card carta);  
}
```

3. Parser, Tokenizer, Tokens: A pesar de ser tres clases diferentes, no es posible dar una explicación de cada una por separado, ya que funcionan como un todo. El Parser es responsable de construir un árbol de sintaxis a partir del conjunto de tokens creado por el Tokenizer. Se usa para analizar el código entrante y construir un árbol de sintaxis abstracta (AST) que luego se puede usar para generar código intermedio. El Tokenizer lee el código fuente de entrada, realiza un análisis léxico y crea una secuencia de tokens en función de su interpretación del texto. Cada token tiene un significado semántico y contiene información sobre qué tipo de símbolo representa. La clase Tokens hace coincidir los símbolos de los tokens con sus identificadores y valores correspondientes. Esto es importante porque le permite al intérprete comprender el contexto de lo que se está analizando. Esto permite construcciones de lenguaje más avanzadas, como flujo de control, bucles y funciones. Además, esto también admite extensiones de lenguaje u operadores que se pueden agregar para ampliar su conjunto de características sin necesidad de volver a escribir ninguna pieza de código existente.



Las cartas pueden ser añadidas al juego de dos formas, manualmente mediante un archivo llamado Cartas.txt en el cual se encuentran todas las cartas del juego o a través de un menú interactivo que permite la creación de cartas. Cuando se hace de forma manual es necesario seguir una serie de estructuras lógicas con el fin de tener una misma base para todas las cartas, en este caso lo hacemos de la siguiente forma:

Nombre+[Descripción]+ poder # + facción #+ que (Efectos a realizar: pueden ser cuantos desee el usuario e incluso aplicarle condiciones). Ejemplo de esto encontramos la siguiente carta:

(Vampiro: garkain) [Los bebedores de sangre y los comedores de cadáveres son tan repugnantes que su misma fealdad paraliza a los enemigos... aumenta el poder de las cartas Monster aliadas en 1 cuando tengan menos power que 5] poder 4 facción 1 que SubePoder 1 cuando MenosPoderQue 5 facción 1

```
(John Matais) [Esta plaza debe llevar los nombres de mis soldados, de los muertos. No el mío... disminuye en 5 el power de las cartas Scoria] poder 1 facción 2 que QuitePoder 5 cuando facción 4  
(Esterad Thysen) Como todos los hombres de Thysen, era alto, de constitución poderosa y criminalmente guapo... disminuye en 2 el power de las cartas Scoria con power mayor a 5] poder 4 facción 2 que QuitePoder 2 cuando facción 4 MasPoderQue 5  
(Vivian) [Aumenta el poder de las cartas Neutral con power menor a 3] poder 2 facción 3 cuando MenosPoderQue 3 cuando MasPoderQue 3  
(Philippa Eilhart) [Pronto el poder de los reyes se marchitará y la Logia tomará el lugar que le corresponde... Aumenta en 2 el power de las cartas Neutral con power inferior a 2] poder 2 facción 2 que SubePoder 3 cuando MenosPoderQue 2 facción 2  
(Gerald de Rivia) [Si eso es lo que hace falta para salvar el mundo, es mejor dejar que sucumbe... disminuye en 3 el power de las cartas Neutral aliadas cuando estas tengan power mayor a 7] poder 10 facción 2 que SubePoder -3 cuando MasPoderQue 7 facción 2  
(Catapulta) Los dioses ayudan a los que tienen mejores catapultas... aumenta el poder de las cartas Neutral aliadas en 1 cuando tengan menos power que 5] poder 4 facción 2 que SubePoder 1 cuando MenosPoderQue 5 facción 1
```

Figura 3: Fragmento del archivo cartas.txt



Figura 4: Menú interactivo de creación



2.3. Visual

1. Campo_Juego
2. Crear_Cartas
3. Form1
4. Program
5. Reglas_Juego
6. Seleccion_Cartas

Se utilizó Windows Forms para la sección visual del TCG porque es un poderoso marco de interfaz de usuario, que tiene la capacidad de crear interfaces de usuario complejas y sofisticadas con un mínimo esfuerzo. También proporciona una amplia biblioteca de componentes y controles, para que los desarrolladores puedan crear rápidamente una interfaz interactiva confiable y fácil de usar. La naturaleza de arrastrar y soltar de Windows Forms facilita la creación rápida de elementos gráficos y ofrece un flujo de trabajo más intuitivo que otras tecnologías.

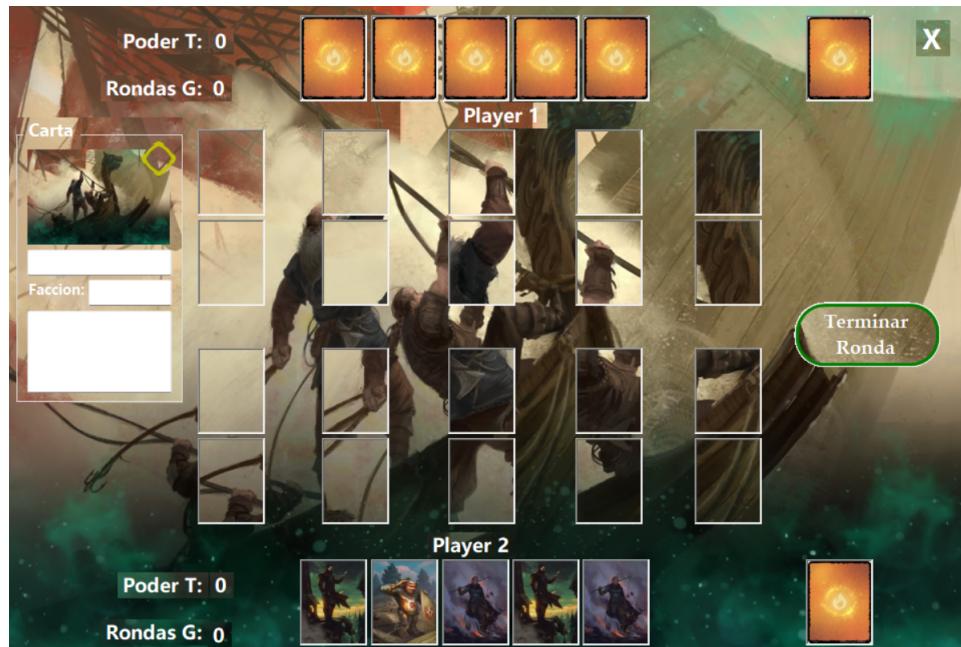


Figura 5: Partida de The Clam Boat

Alejandro Lamelas Delgado Alejandro Ramirez Trueba
@Artagos @AleTheCreation

Mauro Eduardo Campver Barrios
@Mauro - 02

12 de enero de 2023