**GWENT ++**

Gwnet++ es la segunda fase del gwent, donde nos adentramos en la tarea de crear un minicompilador de cartas y efectos.

**¿Qué es un compilador?**

Un compilador es un tipo especial de software que traduce el código fuente escrito en un lenguaje de programación (el lenguaje fuente) a otro (el lenguaje de destino). Los compiladores constan de varias fases.

**Lexer**

En esta etapa se escanea el código fuente que se pretende compilar y se transforma en token. Un token no es más que un contenedor de información donde se guarda, él o la secuencia de caracteres que representan, su posición en el código y el tipo.

Para ello se modeló una clase token que guarda los valores antes expuestos, y en el escáner se utiliza un diccionario que mapea, una expresión regular (las expresiones regulares permiten reconocer patrones dentro de un string) con su respectivo tipo de token. Este escáner se ejecuta a través de un método **Tokenize** definido en la clase del Lexer. Dicho método al final devuelve una secuencia de tokens, descartando todos los espacios en blanco que encuentra.

**Parser**

El parser recibe todo el flujo de tokens que se obtiene luego de pasar la etapa de análisis léxico y se procede a analizar la sintaxis del código, es decir que el flujo de tokens cumple con las reglas del lenguaje que se quiere modelar. Además, se crea el árbol de sintaxis abstracta (AST), es decir los tokens son convertidos en clases del lenguaje base, en este caso C# y representados en forma de árbol. En este caso la cúspide del árbol serían las clases que representan las cartas y los efectos.

Todas las clases que conforman el árbol heredan de una clase ASTNode. Aquí se implementaron una serie de métodos para crear todos los nodos.

**ParseCard:** Cada vez que se encuentra la palabra reservada card en el flujo de tokens se llama a este método, el cual devuelve un nodo carta. La carta a su vez contiene una serie de propiedades, es por ello que dentro de la ejecución se hace un llamado a **ParseCardProperties** que recibe la carta que se acaba de crear y va seteando todas sus propiedades.

Las propiedades de las cartas son principalmente textos y en el caso del Power, un double. Cuando se parsea el power se llama a parsear una operación aritmética, puesto que el poder no tiene que ser representado necesariamente con un número directamente. El método principal para operaciones aritméticas es **ParseTerm** que hace también llamados a otros métodos para verificar el orden en que se ejecutan las operaciones aritméticas.

Para la propiedad OnActivationde las cartas se llama al método **ParseOnActivation** que guarda en una lista todos los efectos que se ejecutarán una vez la carta esté en el campo.

**ParseEffect:** Como su nombre lo indica con este método se parsean todas las declaraciones de efectos. Los efectos están conformados por un nombre, unos parámetros y un Action Body. Para parsear el ActionBody del efecto se llama al método **ParseAction** que lo que hace es devolver una lista de instrucciones que se ejecutarán una vez que se active el efecto. Para ello se implementaron métodos para parsear nodos while y for que representan bucles, también para variables que se pueden declarar dentro del Actionbody y propiedades.

**Chequeo Semántico**

En la fase de chequeo semántico se realizan una serie de comprobaciones dentro de los nodos para verificar que el código tenga sentido, para ello se declaró un método **CheckSemantic** dentro de la clase ASTNode, que obliga a que todos los nodos hijos tengan que implementarlo. Por ejemplo, dentro de la clase carta se comprueba que el power sea un valor numérico, que el range sea válido, o que dentro del source del selector del OnActivation se declare un valor admitido. Aquí también se comprueba que no haya cartas repetidas o que al hacer referencia a alguna variable esté previamente declarada.

**Evaluación**

En la fase de evaluación se ejecutan todas las acciones que representan cada uno de los nodos, para ello se declaró al igual que con **CheckSemantic** un método **Evaluate** dentro de la clase ASTNode, que obliga a que todos los nodos hijos lo implementen. Por ejemplo, dentro del Evaluate de la clase **Card** se crean las cartas dentro del Gwent, o en el caso de la clase **Effect** se ejecuta la lista de instrucciones previamente declarada, es decir el **Evaluate** de Effect hace un llamado al **Evaluate** de cada una de las instrucciones.