BattleCards

Hivan Cañizares Diaz'
Carlos Mauricio Reyes Escudero
C - 111

Contents

1 Creador de Cartas

1.1 Ideas basicas

Para la creacion de las cartas se ha creado un mini-compilador q esta formado por un Parseador en 'Parser.cs' y una jerarquia de clases contenida en 'NodeTree.cs'

1.1.1 Primer paso: Lectura de los archivos con el 'codigo'

La carpeta decks contiene adentro otras carpetas...las cuales...pasaran a constituir los decks, cuyas cartas seran los archivos '.txt' que estas contienen.

1.1.2 Segundo paso: El texto es interpretado

Cada archivo tiene el codigo q es Parseado y pasan a formar un conjunto de instrucciones que al ejecutarse modifican las cartas de acuerdo con las reglas del lenguaje, de manera q estas son corridas en primera instancia para la creacion de la carta y muchas son devueltas para usar durante el juego.

1.1.3 Tercer paso: Los archivos son almacenados en clases

Los archivos se guardan para ser seleccionados antes de comenzar la partida.

1.2 Instrucciones y Expresiones

1.3 Instrucciones

Del codigo solo se obtienen instrucciones, y son estas las q luego evaluan expresiones.

1.3.1 Instruccion 'set'

se escribe de la forma

set 'variable1' 'variable2' to 'expresion'

y asigna a las variables enumeradas el valor resultante.

1.3.2 Instruccion 'if'

Escrita de la forma

if expresion
start
instruccion1
instruccion2

else
instruccion3
instruccion4
end

la cual evalua la expresion y en funcion de ello ejecuta las instrucciones 1,2 o 3,4 ,notese el start-end para englobar el cuerpo del if.

1.3.3 Instruccion 'while'

El while evalua como un if y se mantiene ejecutandose mientras la expresion retorne verdadera.

1.3.4 Instruccion 'for'

Se escribe de la manera

for expresion1 to expresion2
start
instruction1
instruction2
end

recorriendo todos los valores enteros entre ambas expresiones y ejecutando las instrucciones correspondientes.

1.3.5 Instruccion 'def'

Escrita de la forma

def 'ActionName' 'int' 'descripcion'
start
'cuerpo de la accion'
end

genera una accion la cual es asignada a la carta y puede ser invocada durante el transcurso del juego(Una lista de acciones por defecto son asignadas a todas las cartas y estas se encuentran en un documento llamado 'defaultactions.txt').

1.4 Expresiones

Las expresiones son evaluadas en las instrucciones y determinan el comportamiento de las mismas de acuerdo a cuanto evaluan.

1.4.1 Expresion 'Numero'

Esta expresion evalua en un numero directamente.

1.4.2 Expresion 'Variable'

Esta expresion evalua un texto, q llega a su valor numerico de acuerdo a su valor en un diccionario.

1.5 Expresiones 'Binarias'

Estas generalizan cada posible expresion binaria en una clase, dividiendose por operadores:

1.5.1 Algebreicas

Adicion(+) Esta operacion suma las expresiones evaluadas a la izquierda y la derecha.

Sustraccion(-) Esta operacion resta las expresiones evaluadas a la izquierda y la derecha(no es conmutativa).

Multiplicacion(*) Esta operacion multiplica las expresiones evaluadas a la izquierda y derecha.

Division(/) Esta operacion Divide las expresiones evaluadas a la izquierda y derecha.

1.5.2 Booleanas

Las expresiones booleanas se evaluan como true(valor>=0) y false(valor<0):

Mayor(>) Esta operacion resta las expresiones.

Menor(<) Esta operacion suma las expresiones.

Igual(=) Esta operacion suma y resta las expresiones y devuelve el minimo.

OR(|) Esta operacion devuelve el maximo de las expresiones.

AND(&) Esta operacion devuelve el minimo de las expresiones.

Existencia(self.exist(),adv.exist()) Comprueba la existencia de alguna carta q cumple con algun valor en un campo.

Totalidad(self.all(),adv.all()) Comprueba que todas las cartas de un campo cumplen alguna condicion.

1.6 Ejemplo de codigo 'God.txt'

```
set Life to 100
set Defense Speed to Life/8
set Attack to Life

if 2=2&1<3
start
set Life to 1000
end

while Life>100
start
set Life to Life/3
end

set Life to Life+100
set Attack to Attack-70

set adv.Life to 666

def Cataclism 1
start
if self.exist(Life>600)
start
set adv.Life to 0
end
end
```

2 Estructura del juego

2.1 Clase 'Card'

La clase card es la clase central del juego, esta consiste de un nombre de la carta, un diccionario con las estadísticas y su respectivo valor y una lista de acciones que pueden ejecutar. El juego gira en torno a las estadísticas de las cartas y sus acciones.

2.2 Inicio del juego

2.2.1 Presentación del juego

El juego se ejecuta en el archivo Program que está en la carpeta principal del proyecto. Este archivo da la bienvenida al jugador y llama al método MainMenu de la clase Game al final.

2.2.2 Main Menu

El método MainMenu de la clase Game recibe una lista de Deck (Deck es un objeto en el que se encuentra la lista de cartas con la que se va a jugar). Este método es el encargado de recopilar todos los mazos de cartas que han sido creados y después hechar a andar el juego. Se enseñarán unas opciones al inicio qué son "Build Decks" y "Exit" cuando el método es llamado desde Program con una lista de Deck vacía para que continue la ejecución el usuario debe escoger Build Decks. Al escoger esta opción se llamará al método Recopilatory que es el encargado de crear todos los mazos de cartas del juego y luego se llama al método MainMenu nuevamente con el resultado de este método como argumento. Si existen mazos de cartas creados ahora además de las dos opciones anteriores aparecera la opción "Start Game" qué pasará a la siguiente fase de ejecución del programa con los mazos creados hasta el momento llamando al método Start.

2.2.3 Start

Al llamar este método se entra en un bucle en el que se le pide al usuario que cree a los jugadores, determinando el deck que utilizará, el nombre del jugador y si es un jugador virtual o no. Luego de obtener esta información se crean los jugadores y con estos el tablero de juego. Del que se llama al método FirstTurn que da inicio a la partida.

2.2.4 First Turn

En el primer turno de juego no se realizan ataques por lo que en este momento solo se invocan las primeras cartas, luego se llama al método PLAY de el primer jugador y en el método PLAY se llama al método PLAY del jugador contrario y así sucesivamente hasta el final de la partida. El método Play devuelve el nombre del ganador y cuando se devuelve a la primera llamada se muestra el nombre del ganador y luego se inicia el juego nuevamente.

2.3 Flow del juego

2.3.1 Clase 'Board'

La clase Board tiene como atributos el Player1 y el Player2, en estos Player se encuentran todos los datos del juego.

2.3.2 Clase 'Player'

La clase Player consta de el nombre de el jugador, el campo de el jugador y su mazo de cartas. En el método PLAY primero se llama a la invocación de las cartas y después de que las invocaciones hayan sido realizadas se realizan las acciones de cada carta pidiendole al usuario que seleccione las acciones para cada carta y ejecutandose al momento, cada acción es interpretada utilizando el compilador con un método que recibe ambas cartas sobre las que se le realiza una acción.

2.3.3 Clase 'VirtualPlayer'

La clase Virtual Player hereda de la clase Player y tiene un método override de el método PLAY, así que de esa forma es utilizado como un Player cuando es llamado. El método PLAY de Virtual Player en vez de pedirle las acciones de las cartas al usuario llama al método GetActions de Virtual Player que devuelve un array de AccionIndex que es un objeto que contiene 3 indexes que sirven para indicar la carta a la que esta dirigida la accion y la accion a ejecutar. A cada carta de el campo se le llama el método GetCardAccion que cicla por todas las posibles acciones de la carta y las ejecuta en un campo simulado contra cada carta del campo, el método devolverá la jugada con mayor valor. El valor de una jugada se calcula según el valor del tablero usando el método MeanValues que suma todas las estadísticas de las cartas del campo propio y les resta la estadísticas de las cartas del contrario a cada estadística se le multiplica una constante que representa cuan importante es una estadística respecto a las demás para este jugador virtual el valor que se devuelve es la suma de todas las diferencias de las estadísticas. Una vez que cada carta tiene sus acciones definidas son ejecutadas en el método PLAY y mostradas en consola a medida que se ejecutan.

2.4 Fin del juego

El método End devuelve un bool que determina si terminó la partida o no esto se comprueba en el método PLAY y en caso de que sea true se devuelve el ganador y se reinicia el juego.