Relazione progetto **Programmazione Avanzata JLOGO**

Nome: Chiocchi Francesco
Matricola: 105504
Data:26/09/2022

BREVE INTRODUZIONE

Ho definito una grammatica per il linguaggio di programmazione **LOGO** (Commands.g4) grazie al tool ANTLR4 (ANother Tool for Language Recognition), che è un potente generatore di parser per leggere, elaborare, eseguire o tradurre testo strutturato o file binari.

Ho esteso poi la classe generata in automatico da ANTLR4 **CommandsBaseListener** in **LogoBaseListener** dove, scansionando l'intero albero di parsing, ho implementato tutte le istruzioni riconosciute dalla grammatica Commands.g4.

Nonostante la specifica richiedesse "Quando **tre linee** (consecutive ed adiacenti) si chiudono viene individuata un'area chiusa", ho comunque gestito la possibilità che più linee (e.g. quattro, cinque…) potessero formare un'area chiusa e quindi un poligono, invece che formare solamente un triangolo (bastava mettere un controllo sul numero dei tratti e poi eliminare dalla lista il primo tratto che non formava un'area chiusa e quindi un triangolo).

Alcuni file di esempio e relativi risultati sono disponibili nella cartella "**Esempi**" presente nel progetto.

Infine, nel progetto Gradle, è presente un file in formato PDF, chiamato **Istruzioni I-O**, che descrive come passare in input un file contenente la grammatica LOGO (quindi le relative istruzioni) e dove viene salvato il corrispondente file di output se l'utente non specifica un percorso ma solo il nome del file.

Assegnamento delle RESPONSABILITA’

Panel: ha la responsabilità di conoscere: tiene traccia di tutte le caratteristiche generali di un piano geometrico (e.g. figure presenti nel piano o posizione attuale del cursore).

Cursor: ha la responsabilità di conoscere: tiene traccia se sta disegnando o meno, del colore, della dimensione della figura generata dopo uno spostamento e della direzione.

Position: rappresenta un punto in un piano bidimensionale, quindi ascisse e ordinate.

Color: ha la responsabilità di mantenere tutte le informazioni di un colore, in questo caso RGB, composto da una terna di valori interi che vanno da 0 a 255.

BasicShape: ha la responsabilità di conoscere: tiene traccia del punto di inizio e del punto di fine della figura (e.g. Line, Edge, Curve…) generata dal cursore e il relativo colore.

ClosedArea: ha la responsabilità di conoscere tutte le figure basi che la compongono e il relativo colore.

IFileProgramReader: ha la responsabilità di leggere un programma Logo da un file preso in input dall’utente.

IFileProgramWriter: ha la responsabilità di scrivere il risultato dell’esecuzione del programma su un file.

ISavingFile: ha la responsabilità di salvare il file dopo che è stato scritto.

Commands.g4: è una grammatica che ha il compito di riconoscere i vari token presenti nel file da leggere, e quindi di riconoscere le istruzioni scritte nel linguaggio di programmazione LOGO.

LogoBaseListener: ha la responsabilità di eseguire tutte le istruzioni LOGO presenti nel file da leggere (dopo che la grammatica le abbia riconosciute come tali).

Logo (CLI): ha la responsabilità di gestire l'interazione con l'utente tramite la Command Line Interface.

LogoFX (GUI): ha la responsabilità di gestire l'interazione con l'utente tramite un'interfaccia grafica.

LogoController (GUI): ha la responsabilità di gestire gli **eventi** che l'utente seleziona tramite l'interfaccia grafica.

IMPLEMENTAZIONI

Panel:

-SimplePanel

Cursor:

-SimpleCursor

Position:

-Point

Color:

-RGBColor

BasicShape:

-Line

-Future class Edge and Curve…

ClosedArea:

-Polygon

-Future class Circle…

IFileProgramReader:

-FileProgramReader

IFileProgramWriter:

-FileProgramWriter

ISavingFile:

-SavingFile

CommandsBaseListener:

-LogoBaseListener

È POSSIBILE AGGIUNGERE FACILMENTE NUOVI COMANDI?

Per l'aggiunta di nuovi comandi, basterà semplicemente aggiungere alla grammatica **Commands.g4** i nuovi comandi, ridefinendo la regola instruction presente nella grammatica, e rigenerare tutti i file in modo automatico, tramite l'opzione "Generate ANTLR4 Recognizer", che mi consentono di fare delle operazioni riguardanti l'albero di parsing.

È POSSIBILE AGGIUNGERE NUOVE FIGURE AL MIO PROGETTO LOGO?

Si, è possibile, basta creare una nuova classe (e.g. Edge, Curves…), fare in modo che estendi la classe astratta **BasicShape** (tramite ovviamente extends), nella quale, indipendentemente dal tipo di figura base (linea, curva…), verrà definito un punto di partenza e un punto di arrivo e informazioni aggiuntive come colore e grandezza;

Per quanto riguarda invece l'area chiusa, abbiamo che le nuove figure basi aggiunte, magari, possono formare un'area chiusa diversa dal Poligono, come il cerchio. In questo caso quindi, basterà creare una nuova classe Circle che estendi l'interfaccia **ClosedArea** e avrà tutti i metodi che servono per poterla rappresentare nel disegno. Ovviamente, per fare questo nella lista contenente le figure basi dell'area chiusa, non ho limitato il numero di figure basi a tre (il minimo per rappresentare un poligono), ho semplicemente detto che la lista passata in input non deve essere vuota, perché nel caso di un arco, per rappresentare un'area chiusa potrebbe bastare un solo elemento.

ISTRUZIONI

Test

C:\...**gradle test**

$ ./ **gradlew test**

Build

C: \...**gradle build**

$ ./ **gradlew build**

Cli execution

C:\...**gradle --console plain :cli:run**

$ ./ **gradlew --console plain :cli:run**

Da progetto Gradle**: ToolWindows/Gradle/JLogo/cli/Tasks/application/run**

GUI

C:\...**gradle :gui:run**

Da progetto Gradle**: ToolWindows/Gradle/JLogo/gui/Tasks/application/run**

INFORMAZIONI AGGIUNTIVE

-Si può vedere il progetto anche su GitHub al seguente link: JLogo

-Si può leggere il file README.md per gli sviluppi di base, implementazione media e avanzata del progetto.

- Si possono vedere le istruzioni per la linea di comando nel file “Istruzioni I-O” presente in JLogo.