AKADEMIA NAUK STOSOWANYCH W TARNOWIE

Konwerter trajektorii z wykorzystaniem ANTLR

Akademia Nauk Stosowanych w Tarnowie
Katedra Informatyki

Kierunek: Informatyka
Przedmiot: Języki formalne i kompilatory II
Semestr: letni 2022
Rok: III

Wykonawcy
Dominik Pająk
Jakub Serwin
Robert Małek

Prowadzący
dr hab inż. Radosław Klimek

1. Zastosowane formaty trajektorii

W projekcie użyliśmy dwóch prostych zapisów trajektorii opartych na liście składającej się z

- liczby porządkowej (ID)
- obecnej klatki (FR)
- koordynatów położenia (X,Y,Z)
- koordynatów półosi (A,B)
- kąta nachylenia (ANGLE)
- koloru trajektorii (COLOR)

Zaproponowana budowa obiektu trajektorii została zainspirowana standardem zainspirowanym przez JuPedSim Open sourcowy program do symulacji, analizowania oraz analizy Pedestran Dynamic. (https://www.jupedsim.org/jpscore_trajectory.html).

Pierwszy model trajektorii wygląda w następujący sposób:

LP	FR	X	Y	Z	A	B	ANGLE	COLOR
1	0	3.30	3.33	0.00	0.18	0.25	-90.00	0
2	0	4.50	4.44	0.00	0.18	0.25	-90.00	0
3	0	3.60	3.70	0.00	0.18	0.25	180.00	0
4	0	3.60	4.07	0.00	0.18	0.25	180.00	0
5	0	4.50	4.07	0.00	0.18	0.25	-90.00	0
6	0	4.20	3.33	0.00	0.18	0.25	-90.00	0

Następny został delikatnie przekształcony, wzorując się strukturą listy Set

```
[
{1, 0, 3.30, 3.33, 0.00, 0.18, 0.25, -90.00, 0},
{2, 0, 4.50, 4.44, 0.00, 0.18, 0.25, -90.00, 0},
{3, 0, 3.60, 3.70, 0.00, 0.18, 0.25, 180.00, 0},
{4, 0, 3.60, 4.07, 0.00, 0.18, 0.25, 180.00, 0},
{5, 0, 4.50, 4.07, 0.00, 0.18, 0.25, -90.00, 0},
{6, 0, 4.20, 3.33, 0.00, 0.18, 0.25, -90.00, 0}
]
```

Zostały stworzone dwie osobne gramatyki do parsowania koordynatów. Na ich podstawie wygenerowano Listener, Visitor oraz Parser

2. Gramatyka pierwszej struktury trajektorii

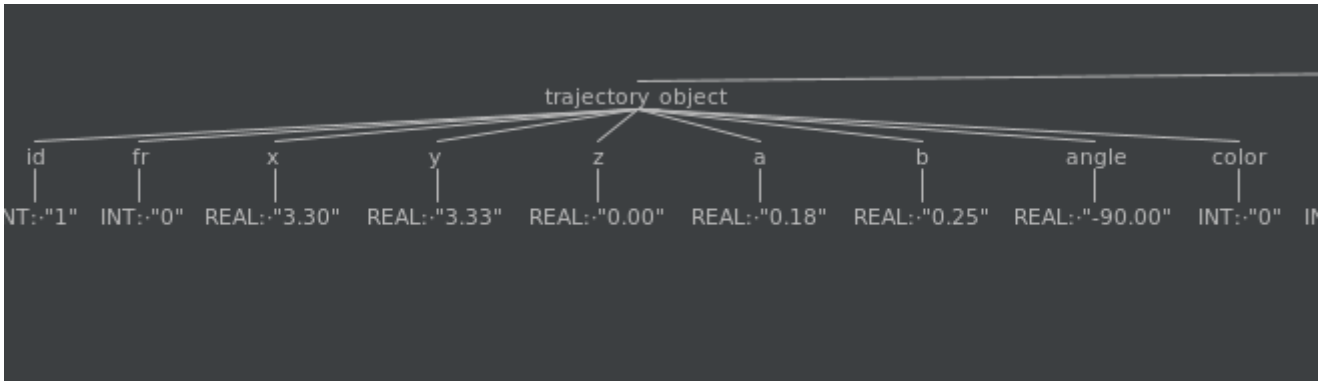
```
// Grammar for Trajectory .traj file format
grammar JpsCoreTrajectoryGrammar;

@header {
   package antlr;
}

jps_core_trajectory_grammar: (trajectory_list) EOF;

/*
* Lexer Rules
*/

fragment COMMENT    : '#' (~[\n\r] .)* ;

trajectory_list : (trajectory_object)*;

trajectory_object : id fr x y z a b angle color;

// Token
id      : INT;
fr      : INT;
x       : REAL;
y       : REAL;
z       : REAL;
a       : REAL;
b       : REAL;
angle   : REAL;
color   : INT;
BREAKPOINT: '\t' -> skip;
WS : [ \t\n] -> skip;

INT
: '0'
| [1-9] [0-9]*
| '-' [1-9] [0-9]*
;


REAL
: INT
| [0-9]* '.' ([0-9])*
| '-' [0-9]* '.' ([0-9])*
;

NEW_LINE : '\r'? '\n' -> skip;
```

3. Gramatyka drugiej struktury trajektorii

```
// Grammar for Trajectory .traj file format
grammar JsonTrajectoryGrammar;

@header {
   package antlr;
}

json_trajectory_grammar: (trajectory_list)? EOF;

/*
* Lexer Rules
*/

fragment COMMENT    : '#' (~[\n\r] .)* ;

trajectory_list : LIST_START (trajectory_object)* LIST_END;

trajectory_object : ROW_START id BREAKPOINT fr BREAKPOINT x BREAKPOINT y
BREAKPOINT z BREAKPOINT a BREAKPOINT b BREAKPOINT angle BREAKPOINT color
ROW_END;

// Token
id      : INT;
fr      : INT;
x       : REAL;
y       : REAL;
z       : REAL;
a       : REAL;
b       : REAL;
angle   : REAL;
color   : INT;
BREAKPOINT: ',';
WS : [ \t\n] -> skip;
LIST_START: '[';
LIST_END: ']';
ROW_START: '{';
ROW_END: '}' | '},';

INT
: '0'
| [1-9] [0-9]*
| '-' [1-9] [0-9]*
;


REAL
: INT
| [0-9]* '.' ([0-9])*
| '-' [0-9]* '.' ([0-9])*
;

NEW_LINE : '\r'? '\n' -> skip;
```

4. Testowanie gramatyki

Drzewo dla pierwszej wersji zapisu trajektorii



Fragment drzewa dla drugiej wersji zapisu trajektorii