Syntax of Chiron Language

A Chiron program is a collection of statements and the program is saved in a file with `.tl` extension. The grammar of Chiron language can be found at `ChironCore/turtparse/tlang.g4`. You are encouraged to take a look at the grammar file to better understand the syntax of Chiron language. However, we have sufficient examples in this document to help you understand the syntax of Chiron language.

1 VARIABLE NAMING CONVENTION

A variable has the naming convention shown by the regular expression: `:[a-zA-Z\_] [a-zA-Z0-9]*`. As you can see the variable name always starts with a clon (:) symbol. For example, `:vara10` is a valid variable name, however `_vara` is not.

2 STATEMENTS

Chiron programming language support the following types of statements. In Chiron language a block of statements is enclosed by square brackets unlike the curly braces in C.

(1) Assignment statement
(2) If statement
(3) If-else statement
(4) Repeat statement
(5) Move statement
(6) Pen statement
(7) Goto statement

2.1 Assignment Statement

In an assignment statement you can only assign arithmetic expression or a constant value to a variable.

$$\langle variable \rangle := \langle expression \rangle$$

We show some examples below,

- `:move = 20`, assigns a constant value `20` to variable `:move`
- `:x = :y + :z`, evaluates `:x + :y` and assigns it to variable `:x`

2.2 If Statement

$$if\ (\langle condition \rangle)[\langle statement(s) \rangle]$$

```
if (:x > 10) [
        :x = :x - 1
]
```

List. 1. if `:x` is greater than 10 then execute (`:x = :x -1`) else do nothing.

2.3 If-else Statement

$$if\ (\langle condition \rangle)[\langle statement(s) \rangle]\ else\ [\langle statement(s) \rangle]$$

```
if (:y1 > 42) [
  :y2 = :z - 20

] else [
  :y2 = :w - 20
  :y1 = :x1 + 20
]
```

List. 2. if :y1 is greater than 42 execute the then branch else execute the else branch.

2.4 Repeat Statement

$$repeat\ \langle value \rangle[\langle statement(s) \rangle]$$

You can implement loop using `repeat` keyword in Chiron language. The ⟨*value*⟩ above can be either a variable name or a constant, any other expression is not supported.

```
repeat 10 [
   :vara = :vara + 1
]
```

List. 3. The body of the `repeat` statement is executed 10 times.

2.5 Move Statements

$$\langle move\_command \rangle = \langle expression \rangle$$

A move command is either of `forward`, `backward`, `left`, `right` and is used to move the turtle sprite on the canvas. Some examples are shown below.

```
:varb = 200
forward :varb
```

List. 4. The turtle sprite moves 200 steps forward.

```
right 90
backward 200
left 10
```

List. 5. The turtle sprite rotates right by an angle 90 degree then moves 200 steps forward and finally rotates left by 10 degree.

2.6 Pen Statements

$$\langle pen\_command \rangle$$

The pen statement here ($\langle pen\_command \rangle$) takes no argument. There are two types of pen statements penup and pendown. pendown tells the interpreter to draw on the canvas when the turtle sprite moves next. penup tells the interpreter not to write anything on canvas on movement of the turtle sprite.

```
pendown
forward 80
penup
forward 20
```

List. 6. For the first forward statement, turtle sprite will move 80 steps forward and while moving it will will draw a line. For the second forward statement, the turtle sprite will move 20 steps forward but this time it will not draw anything since the penup was called just before the forward statement.

2.7 Goto Statement

$$goto(\langle x - coordinate \rangle, \langle y - coordinate \rangle)$$

The goto command moves the turtle sprite from the current position to the given x and y coordinates by following the shortest path.

```
pendown
goto (100, 100)
```

List. 7. This will draw a diagonal line in the direction of the turtle sprite's movement from the origin to the coordinate (100, 100) in the canvas.