Summary

This ‘mini-programming language’ is used to create and modify multiple different autonomous programs, with as little code modification as possible. We designed it to work with the event-driven setup of our autonomous code, which is described in a different document.

An example:

#! Initial robot settings;

INIT, X=11.0, Y=12.5, HEADING=20, ALLIANCE=red;

#! Move backward for 1 second, then turn towards mountain for 0.5 seconds;

STARTBLOCK;

NAVIGATION, EVENT=MoveEvent, TIME=-1000;

NAVIGATION, EVENT=TurnEvent, TIME=500;

ENDBLOCK;

#! Pause for 3 seconds;

STARTBLOCK;

ACTION, EVENT=PauseEvent, TIME=3000;

ENDBLOCK;

#! Move forward for 2 seconds;

STARTBLOCK;

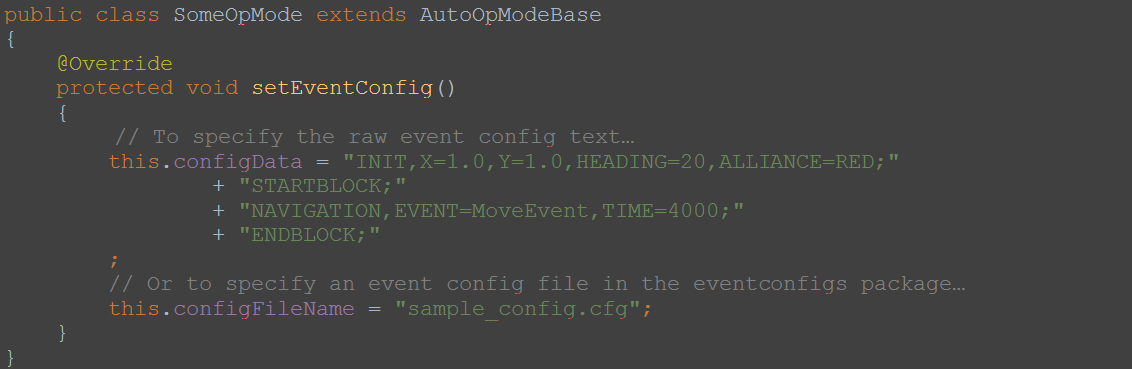
NAVIGATION, EVENT=MoveEvent, TIME=2000;

ENDBLOCK;

Setting Up a New OpMode

The 3 steps to setting up a new autonomous OpMode:

1. Create the event config data, using this documentation as a reference
2. Create the OpMode’s class in the “com.lmrobotics.litcode.autonomous.opmodes” package (See next page for an example)
   1. Make it extend the AutoOpModeBase class
   2. Override the setEventConfig() method
   3. In the setEventConfig() method, set the event config data or file
      1. To put the event config text directly into the class…
         1. this.configData = “Config text goes here…”
      2. To specify a file to load event config data from a file… (**Warning**: this option is not setup yet)
         1. this.configFileName = “file\_name\_with.extension”
3. **This step is very important.** Last you need to add the new OpMode to the OpMode register, so it will show up on the driver station as an available OpMode.
   1. Open the *com.qualcomm.ftcrobotcontroller.opmodes. FtcOpModeRegister* file.
   2. In there, you will see a bunch of these lines: manager.register(“OpMode Name”, com.lmrobotics.litcode.autonomous.opmodes.SomeOpMode.class);
      1. Copy one of these lines, and change OpMode Name to what your OpMode will be called on the driver station, and replace SomeOpMode with your new OpMode’s class name.



General Information

* Event config “tags” are grouped into:
  + Instructions
    - Ex. STARTBLOCK, INIT, NAVIGATION
  + Parameters
    - Ex. X=, EVENT=, TIME=
  + Comments
    - Indicated by starting with “#!”
* All parameters are associated with an instruction tag
* Parameters are used by the event config parser to generate events and other data used by autonomous (like initial position, etc.)
* Instructions and their associated parameters are comma-delimited (separated by commas)
* A set of one instruction indicator and its parameters ends in a semicolon
  + **NOTE:** This includes the comment statements
* Spaces, tabs, newlines and comment statements are ignored by the event config parser, so the instruction-parameter sets can be formatted nicely to be human-readable

Instructions & Parameters

Index

#!

INIT

X=

Y=

HEADING=

ALLIANCE=

STARTBLOCK

ENDBLOCK

NAVIGATION & ACTION

Shared Parameters:

EVENT=

NAVIGATION

…With EVENT=MoveEvent

X=

Y=

TIME=

MAX\_SPEED=

…With EVENT=TurnEvent

HEADING=

TIME=

MAX\_SPEED=

ACTION

...With EVENT=PauseEvent

TIME=

**NOTES**:

* In the following documentation, “#” indicates an integer, “#.#” indicates a double (a number with a decimal portion), “\*\*\*” indicates a string of text, and “|” indicates that the parameter can have one of the possible values given.
* Grid coordinates and headings/directions are relative to the grid we created earlier in development.

#! (Comment Statements)

#! Move forward for 1 second, then turn left for 0.5 seconds;

Comment statements are ignored by the event config parser. They can be used to add descriptions and notes into the event config, just like comments in a normal programming language. Important note: Comment statements must end with a semicolon just like any other instruction-parameter set.

INIT

INIT, X=11.0, Y=12.5, HEADING=20, ALLIANCE=red;

The INIT instruction is used to set various initial settings for autonomous, like where the robot is starting on the field.

Associated Parameters:

X=#.# and Y=#.#

Indicates the starting x and y coordinates of the robot on the field.

HEADING=#

Indicates the angle the robot will initially be facing.

ALLIANCE=blue | red

Indicates the alliance the robot will be starting on. This will soon be replaced by a single pull double throw switch/sensor on the robot.

STARTBLOCK

STARTBLOCK;

The STARTBLOCK instruction indicates that a new block of events is being started, which clears any previous events that have not been added to the event queue with the ENDBLOCK instruction.

Associated Parameters:

None

ENDBLOCK

ENDBLOCK;

The ENDBLOCK instruction creates a new block of events from the events that have been added since the last STARTBLOCK instruction.

Associated Parameters:

None

NAVIGATION and ACTION

The NAVIGATION and ACTION instructions are for creating the events that will go in the the blocks of events. Events will be queued in the order they are placed in the event config. This only applies to events of the same type (NAVIGATION or ACTION), since the navigation and action systems run at the same time during an event block.

Shared Parameters:

EVENT=\*\*\*

What type of action or navigation event to add. This is the exact name of the event’s corresponding class (for example MoveEvent or PauseEvent).

**NAVIGATION**

NAVIGATION, EVENT=MoveEvent, TIME=375, MAX\_SPEED=0.8;

NAVIGATION, EVENT=TurnEvent, TIME=375, MAX\_SPEED=0.8;

The NAVIGATION instruction indicates an event for the navigation system.

Associated Parameters for EVENT=MoveEvent:

X=#.# and Y=#.#

Indicates the target x and y coordinates to travel to, these will be used only when the navigation method using sensor data has been implemented.

TIME=#

TIME is specified for the navigation version without the use of sensors. This will tell the robot to move forward (positive) or backward (negative) for the specified amount of time.

MAX\_SPEED=#.#

The maximum power level or speed the drive motors are allowed to be set to, up to 1.0. The drive motors will be scaled from the default max of 1.0 to this specified value.

Associated Parameters for EVENT=TurnEvent:

HEADING=#

Indicates the direction (angle) the robot should turn to face. This parameter will be used once we have a working heading-angle system.

TIME=#

How long the robot should turn for, used only for time-based turning instead of sensor-based turning. The robot can be told to turn CCW (positive) or CW (negative) by changing the sign on the time value.

MAX\_SPEED=#.#

The motors will not exceed this speed while turning; see the MAX\_SPEED parameter for MoveEvent above for more info.

**ACTION**

ACTION, EVENT=PauseEvent, TIME=1000;

The ACTION instruction indicates an event for the action system.

Associated Parameters with EVENT=PauseEvent:

TIME=#

How long the actions system should pause before moving to the next event. If placed alone in its own block, both the action system and navigation system will be paused.