FRC 3603

Release 0.1

Wile E. Coyote

May 23, 2022

CONTENTS

Welcome to the documentation place for FRC 3603, the Cyber Coyotes. We are a small, but dedicated FRC team in central Michigan. Our aim with this documentation is to share our learning with future team members and the larger FIRST community. If you found this helpful, spotted a mistake, or have a suggestion, please drop us an email CyberCoyotes AT gmail or on the Reed City Robotics Facebook page

CHAPTER

ONE

TABLE OF CONTENTS

1.1 Controls Team

We are creating this section of the handbook to our FTC team coaches, students and high school programmers learn more about programming and controls in the FRC setting.

1.1.1 PID

PID Theory and Robotics Explained video. PYR 3: Tuning PID with Phoenix Tuner video.

1.1.2 Motion Profiling

When controlling a mechanism, is often desirable to move it smoothly between two positions, rather than to abruptly change its setpoint. This is called "motion-profiling," and is supported in WPILib through the TrapezoidProfile class

WPI

For a description of the WPILib motion profiling features used by these command-based wrappers, see Trapezoidal Motion Profiles in WPILib. Note: The TrapezoidProfile command wrappers are generally intended for composition with custom or external controllers. For combining trapezoidal motion profiling with WPILib's PIDController, see Combining Motion Profiling and PID in Command-Based. To further help teams integrate motion profiling into their command-based robot projects, WPILib includes two convenience wrappers for the TrapezoidProfile class: TrapezoidProfileSubsystem, which automatically generates and executes motion profiles in its periodic() method, and the TrapezoidProfileCommand, which executes a single user-provided TrapezoidProfile. Motion Profiling through TrapezoidProfileSubsystems and TrapezoidProfileCommands

CTRE

- Motion Magic and other Motion Profiling with CTRE.
- 254 PDF explaining motion planning
- 2019 Motion Magic slide presentation. Video of Corey Applegate (FRC 3244) CTRE Motion Magic

Come highly recommended Team 195- FRC Motion Control: part 1 BUT… its from 2018 AND its 4.5 hrs long for both parts

Resources

- Motion Profiling - wiki page
- Telemetry docs docs page.
- Control Modes Configuration for TalonFX and Falcon500 ChiefDelphi post.

1.1.3 PathPlanner

- RDc12 - PathPlanner github
- M Jansen4857 github
- Introoducing PathPlanner 2.0 - ChiefDelphi.

1.1.4 Power Distribution Hub

- REV - Power Distribution Hub Overview.
- WPI - Create a PDH Object.

1.1.5 Pixy2

- Pixy2 - wiki.
- Theory Ball Tracking Graph - Enginerds.
- Subsystems - 2021 Skills Bot.
- Vision Subsystem.

1.1.6 Code Examples

Our example code can be found below and at <github.com/CyberCoyotes>. It should be noted that we do not have a mentor with extensive programming knowledge, so programming for Controls is an area that we are still learning a lot.

2021 Rapid React repository.

Examples of other FRC teams

Enginerds 2021 Skills Bot

A lot of helping examples of command based Commands and Subsystems, button assignments, etc from a successful FRC team

Team 1711 - Raptors.

Team 1918 - NCGears.

Sean Sun - FRC 0 to Autonomous.

Helpful tutorials with video and code examples

Creating and Following a Trajectory.

1.1.7 Style Guide

Source: Google Java Style Guide<Google Java Style Guide>

Block indentation: +2 spaces Each time a new block or block-like construct is opened, the indent increases by two spaces. When the block ends, the indent returns to the previous indent level. The indent level applies to both code and comments throughout the block. (See the example in Section 4.1.2, Nonempty blocks: K & R Style.)

4.3 One statement per line Each statement is followed by a line break.

4.4 Column limit: 100 Java code has a column limit of 100 characters. A "character" means any Unicode code point. Except as noted below, any line that would exceed this limit must be line-wrapped, as explained in Section 4.5, Line-wrapping.

Each Unicode code point counts as one character, even if its display width is greater or less. For example, if using fullwidth characters, you may choose to wrap the line earlier than where this rule strictly requires.

Exceptions:

Lines where obeying the column limit is not possible (for example, a long URL in Javadoc, or a long JSNI method reference). package and import statements (see Sections 3.2 Package statement and 3.3 Import statements). Command lines in a comment that may be copied-and-pasted into a shell. Very long identifiers, on the rare occasions they are called for, are allowed to exceed the column limit. In that case, the valid wrapping for the surrounding code is as produced by google-java-format. 4.5 Line-wrapping Terminology Note: When code that might otherwise legally occupy a single line is divided into multiple lines, this activity is called line-wrapping.

There is no comprehensive, deterministic formula showing exactly how to line-wrap in every situation. Very often there are several valid ways to line-wrap the same piece of code.

Note: While the typical reason for line-wrapping is to avoid overflowing the column limit, even code that would in fact fit within the column limit may be line-wrapped at the author's discretion.

Tip: Extracting a method or local variable may solve the problem without the need to line-wrap.

4.5.1 Where to break The prime directive of line-wrapping is: prefer to break at a higher syntactic level. Also:

When a line is broken at a non-assignment operator the break comes before the symbol. (Note that this is not the same practice used in Google style for other languages, such as C++ and JavaScript.) This also applies to the following "operator-like" symbols: the dot separator (.) the two colons of a method reference (::) an ampersand in a type bound (<T extends Foo & Bar>) a pipe in a catch block (catch (FooException | BarException e)). When a line is broken at an assignment operator the break typically comes after the symbol, but either way is acceptable. This also applies to the "assignment-operator-like" colon in an enhanced for ("foreach") statement. A method or constructor name stays attached to the open parenthesis (() that follows it. A comma (,) stays attached to the token that precedes it. A line is never broken adjacent to the arrow in a lambda, except that a break may come immediately after the arrow if the body of the lambda consists of a single unbraced expression. Examples: MyLambda<String, Long, Object> lambda = (String label, Long value, Object obj) -> { … };

Predicate predicate = str -> longExpressionInvolving(str); Note: The primary goal for line wrapping is to have clear code, not necessarily code that fits in the smallest number of lines.

4.5.2 Indent continuation lines at least +4 spaces When line-wrapping, each line after the first (each continuation line) is indented at least +4 from the original line.

When there are multiple continuation lines, indentation may be varied beyond +4 as desired. In general, two continuation lines use the same indentation level if and only if they begin with syntactically parallel elements.

Section 4.6.3 on Horizontal alignment addresses the discouraged practice of using a variable number of spaces to align certain tokens with previous lines.