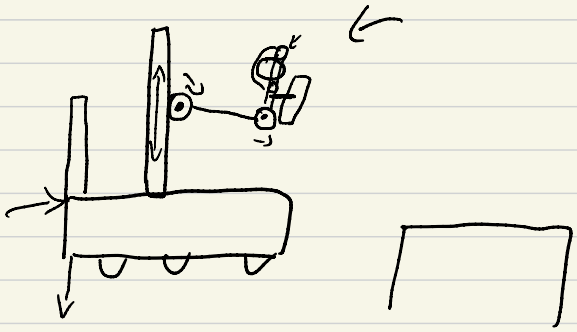
Lecture 2 : Code Walkthrough (Teleop)
Lecture 3 : (Autonomous)

3/5/2023
Peter Wang

Outline

1. Overall software architecture
   - Lib and main
   - Robot.java
   - Constants.java
2. Subsystems
   - Drive.java
   - Elevator.java
   - Elbow.java + Wrist.java
   - Intake.java
   - Limelight.java (Optional)
3. Flywheel + tank drive implementation
4. Autonomous pipelining
   a. Modes
      - Sequential
      - Parallel
   b. Actions (start, update, done, isFinished)
   c. Paths (TrajectoryGenerator.java)
      - Pose2d  (Rigid transform)
      - Rotation2d 
      - Translation2d 
      - Twist2d 
5. Autonomous Execution
6. Guest speaker : Owen Neilson

FRC 2019 game



Structure

1. Lib (For us to use)
2. Robot/Main (Need to implement)

↳ At reset —run→ Main.java → Robot.java

Robot.java :

{ import lib, Subsystems, auto

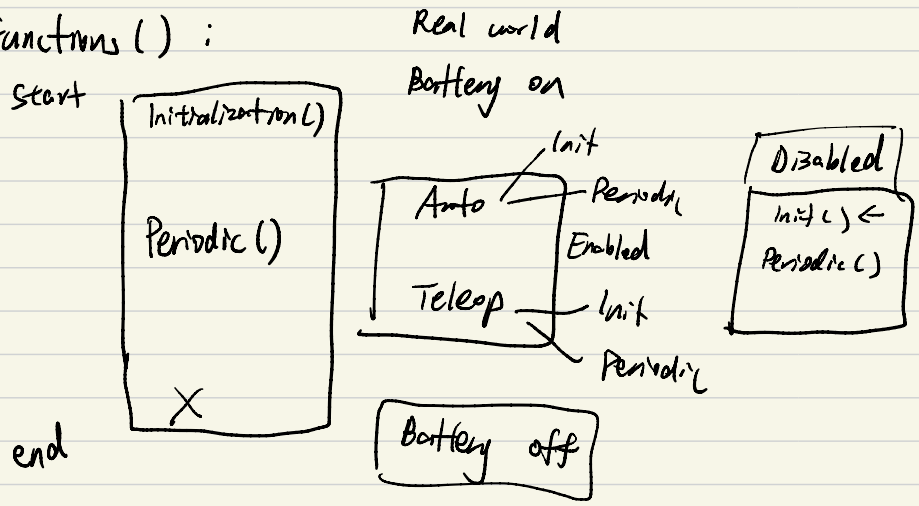
class Robot {

Local variables :

Functions ( )

}

Functions ( ) :



Subsystem

Intake. java

class Intake {

local Variables : SC, Solenoid, etc.

Constructor () {

set up procedure;

Instentiate local variables

}

- Implement inherited functions ()
- Implement custom functions ()
  - Set Power ( )
  - Set Solenoid ( )

}

Modes

A list of instructions to be executed, either sequentially or parallely, or both.

Ex. Path following

Parallel

while set to scoring position

Score / Outtake → + Sequential

Actions

Individual action to take during Auto routine.

Mainly related to a specific action in the subsystem

Ex: In Superstructure:

Set_state(LOW) ⇐
MID
HIGH

set_state_and_outtake() ✗

Java Inheritance:

Interface (skeleton)

Abstract class (skeleton + flesh)

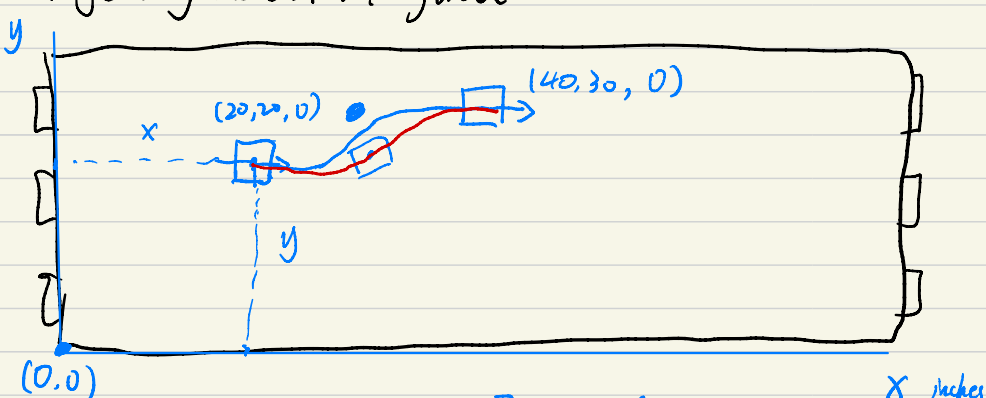
Class (Whole body)

Action :

start() {
  Executed once at the start of the action.
}
update() {
  Executed periodically after start()
}
isFinished() {
  Determines when we should stop update()
}
done() {
  Executed once after update() is terminated.
}

Paths

Trajectory Generator. java



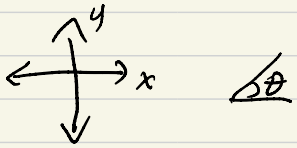
Cartesian

1. Bezier Curve ( )
2. Hermite Spline
   - Cubic x^3
   - Quintic x^5

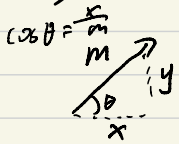
1. Path generation
   * Cubic Hermite spline
     → Starting, mid, Ending position (x, y, θ)

Path Data structure

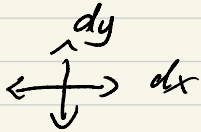
1. Pose2d (x, y, θ)



2. Rotation2d (θ)

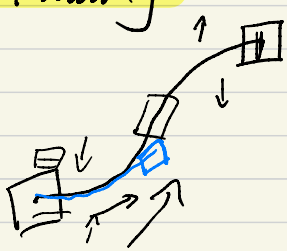


3. Translation2d (dx, dy).



4. Twisted 2d.

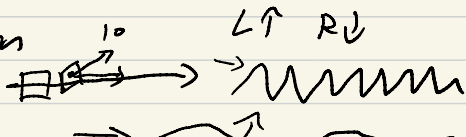
Path Following



Bottleneck
→ Theory ≠ Reality.
↓
Nice plan → Not nice
↓
Reality Gap

Error handling in a closed-loop fashion.

Ways to handle abbreviation



* Bang-bang style
* PID
* Adaptive-Pursuit 2018 in-season. ☐ error
* Nonlinear-Feedback Controller aka RAMSETE
  ☐ low