infinite turtles.
16461

Object Oriented & Command Based Programming in FTC

Introduction

#16461 Infinite Turtles

- 2x North Carolina State Championship Inspire Award Winner
- 2022 World Championship Innovate Award Winner
- 2023 World Championship Division Inspire Award Finalist
- 5th year team

Ryan

- 2023 Software Lead
- 5th Year FTC

Asher

- Software Team
- 1st Year FTC

Why even care?

- Many people see programming in FTC as a **"Means to an end"**

More than robots!

- Job transferable skills!
- Get familiar with programming problem solving skills and structuring code

Performance

- Easily recoverable code
- Easy to make changes
- Fast to debug
- Easier to collaborate
- Less headaches!

What should I consider?

- We'll be going over:
  - Kotlin vs. Java
  - The pitfall of OpMode separation
  - The importance of a "Hardware Map"
  - Modularizing your code
  - Threading?
  - Taking it further: CommandBase
  - Pre-existing solutions, including Nautilus

Kotlin vs. Java

Both Kotlin and Java are JVM-Based languages: They run on the **Java Virtual Machine**

This means both languages can “interop,” or work with each other in the same project.

There’s no true “one better option”

Kotlin

```
val closestPoint = path.closestPoint(pos)

val tangent = closestPoint.tangent
val towards = (closestPoint - pos).normalized

var ratio = pos.distanceTo(closestPoint) * kN

if(closestPoint.x epsilonEquals path.end.x && closestPoint.y epsilonEquals path.end.y) ratio = 1.0

val interp = interpolateVec(tangent, towards, ratio).normalized

var dist = closestPoint.distanceAlongPath

return Pose(interp.x, interp.y, path[dist + turnOffset].angle)
```

Java

```
Vector2 closestPoint = path.closestPoint(pos)

Vector2 tangent = closestPoint.getTangent()
Vector2 towards = closestPoint.sub(pos).normalize()

double ratio = pos.distanceTo(closestPoint) * kN

if(epsilonEquals(closestPoint.getX(), path.end.getX()) && epsilonEquals(closestPoint.getY(), path.end.getY()))
  ratio = 1.0

Vector2 interp = interpolateVec(tangent, towards, ratio).normalize()

double dist = closestPoint.getDistanceAlongPath()

return new Pose(interp.getX(), interp.getY(), path.get(dist + turnOffset).getAngle())
```

Kotlin vs. Java

Kotlin

```kotlin
val closestPoint = path.closestPoint(pos)

val tangent = closestPoint.tangent
val towards = (closestPoint - pos).normalized

var ratio = pos.distanceTo(closestPoint) * kN

if(closestPoint.x epsilonEquals path.end.x && closestPoint.y epsilonEquals path.end.y) ratio = 1.0

val interp = interpolateVec(tangent, towards, ratio).normalized

var dist = closestPoint.distanceAlongPath

return Pose(interp.x, interp.y, path[dist + turnOffset].angle)
```

Java

```java
Vector2 closestPoint = path.closestPoint(pos)

Vector2 tangent = closestPoint.getTangent()
Vector2 towards = closestPoint.sub(pos).normalize()

double ratio = pos.distanceTo(closestPoint) * kN

if(epsilonEquals(closestPoint.getX(), path.end.getX()) && epsilonEquals(closestPoint.getY(), path.end.getY()))
  ratio = 1.0

Vector2 interp = interpolateVec(tangent, towards, ratio).normalize()

double dist = closestPoint.getDistanceAlongPath()

return new Pose(interp.getX(), interp.getY(), path.get(dist + turnOffset).getAngle())
```

Common Pitfalls

```java
@TeleOp
public class GamerOp extends OpMode {
    public DcMotor LeftFront  = null;
    public DcMotor LeftRear   = null;
    public DcMotor RightFront = null;
    public DcMotor RightRear  = null;

//------------------------------InitLoop------------------------------------------------------------\\

    @Override
    public void init() {

//------------------------------PhoneHardWareMap----------------------------------------------------\\

        LeftRear    = hardwareMap.dcMotor.get (" BackLeft   ");
        LeftFront   = hardwareMap.dcMotor.get (" FrontLeft  ");
        RightFront  = hardwareMap.dcMotor.get (" FrontRight ");
        RightRear   = hardwareMap.dcMotor.get (" BackRight  ");

//------------------------------Direction-----------------------------------------------------------\\

        LeftFront .setDirection(DcMotorSimple.Direction.REVERSE);
        LeftRear  .setDirection(DcMotorSimple.Direction.REVERSE);
        RightFront.setDirection(DcMotorSimple.Direction.FORWARD);
        RightRear .setDirection(DcMotorSimple.Direction.FORWARD);

        LeftFront .setMode(DcMotor.RunMode.RUN_WITHOUT_ENCODER);
        LeftRear  .setMode(DcMotor.RunMode.RUN_WITHOUT_ENCODER);
        RightFront.setMode(DcMotor.RunMode.RUN_WITHOUT_ENCODER);
        RightRear .setMode(DcMotor.RunMode.RUN_WITHOUT_ENCODER);

    }

//------------------------------OpMode--------------------------------------------------------------\\

    @Override
    public void loop() {

//------------------------------DriverController----------------------------------------------------\\

        double Drive  = gamepad1.left_stick_y;
        double Strafe = gamepad1.left_stick_x;
        double Turn   = gamepad1.right_stick_x;

        LeftFront  .setPower( + Drive - Strafe - Turn);
        LeftRear   .setPower( + Drive + Strafe - Turn);
        RightFront .setPower( + Drive + Strafe + Turn);
        RightRear  .setPower( + Drive - Strafe + Turn);

        //telemetry.addData("TicksLF",LeftFront.getCurrentPosition());
        //telemetry.addData("TicksLR",LeftRear.getCurrentPosition());
        //telemetry.addData("TicksRF",RightRear.getCurrentPosition());
        //telemetry.addData("TicksRR",RightRear.getCurrentPosition());

        //telemetry.update();
//------------------------------Intake/Belt---------------------------------------------------------\\
    }

}
```

Common Pitfalls

```java
//----------------------------InitLoop--------------------------------

    @Override
    public void init() {

//----------------------------PhoneHardWareMap-----------------------

        LeftRear    = hardwareMap.dcMotor.get (" BackLeft   ");
        LeftFront   = hardwareMap.dcMotor.get (" FrontLeft  ");
        RightFront  = hardwareMap.dcMotor.get (" FrontRight ");
        RightRear   = hardwareMap.dcMotor.get (" BackRight  ");

//----------------------------Direction-------------------------------

        LeftFront .setDirection(DcMotorSimple.Direction.REVERSE);
        LeftRear  .setDirection(DcMotorSimple.Direction.REVERSE);
        RightFront.setDirection(DcMotorSimple.Direction.FORWARD);
        RightRear .setDirection(DcMotorSimple.Direction.FORWARD);

        LeftFront .setMode(DcMotor.RunMode.RUN_WITHOUT_ENCODER);
        LeftRear  .setMode(DcMotor.RunMode.RUN_WITHOUT_ENCODER);
        RightFront.setMode(DcMotor.RunMode.RUN_WITHOUT_ENCODER);
        RightRear .setMode(DcMotor.RunMode.RUN_WITHOUT_ENCODER);

    }
```

Raw hardware access in the OpMode!

Why is this bad?

Accessing raw hardware in an OpMode makes it easy to accidentally have different functionality in different OpModes.

When doing hardware logic in an OpMode, it doesn't transfer to other OpModes!

Common Pitfalls

```java
    @Override
    public void loop() {

//-----------------------------DriverController-----------------------

        double Drive  = gamepad1.left_stick_y;
        double Strafe = gamepad1.left_stick_x;
        double Turn   = gamepad1.right_stick_x;

        LeftFront  .setPower( + Drive - Strafe - Turn);
        LeftRear   .setPower( + Drive + Strafe - Turn);
        RightFront .setPower( + Drive + Strafe + Turn);
        RightRear  .setPower( + Drive - Strafe + Turn);

        //telemetry.addData("TicksLF",LeftFront.getCurrentPosition());
        //telemetry.addData("TicksLR",LeftRear.getCurrentPosition());
        //telemetry.addData("TicksRF",RightRear.getCurrentPosition());
        //telemetry.addData("TicksRR",RightRear.getCurrentPosition());

        //telemetry.update();
//-----------------------------Intake/Belt----------------------------
    }
```

Drive logic in the main OpMode

Why is this bad?

It's not awful, but in most cases drive logic should be shared between all OpModes as much as possible. The abstraction makes it easy to change hardware!

Why is raw hardware logic in OpModes bad?

Structure

- It can be confusing and annoying to have to copy values like lift tuning between OpModes.
- If you swap out a mechanism on your bot, the transition is cleaner!

Organization

- Having “subsystems,” or individual mechanisms on your robot separate in code is great for organization!

Your own Robot/HardwareMap

A very common first step toward organization for teams is separating all of their hardware initialization into a “**Robot**” or “**HardwareMap**” class.

Your own Robot/HardwareMap

```
public class Hardware {

    //Declare hardware devices
    HardwareMap hwMap;
    public YellowJacket435 fl, fr, bl, br;
    private ElapsedTime period  = new ElapsedTime();

    //Defining utils
    public MecanumDrive mecanum;
    public Odometry odometry;

    ...

    /** Initializes the hardware devices and utils */
    public void init(HardwareMap ahwMap){
        hwMap = ahwMap;

        //Assign Motors to custom motor class
        fl = new YellowJacket435(hwMap, "frontLeft");
        fr = new YellowJacket435(hwMap, "frontRight");
        br = new YellowJacket435(hwMap, "backRight");
        bl = new YellowJacket435(hwMap, "backLeft");

        //...

        //Defining mecanum and odometry
        mecanum = new MecanumDrive(this);
        mecanum.init();
        if(odometry==null) {
            odometry = new Odometry(leftValue, rightValue, horizontalValue);
        }

        //Define the IMU in odometry
        odometry.imu = new IMU();
        odometry.imu.init();
    }
}
```

#16461's hardware map in UG

The Robot class contains all references to "subsystems" and individual hardware devices.

It is shared between **all** OpModes, and created on the initialization of them.

Subsystems & Robots

Robot
- Only place that does raw hardware access other
- than subsystems
- Contains all subsystems
- Is shared between all opmodes

Subsystem
- Encapsulates the functionality of hardware

Lift
- Contains an amount of motors
- Manages the control of the motors to go to a certain height

Drivetrain
- Contains all drive motors.
- Exposes a “drive” method that takes a direction to drive in.

Threading?

Although threading is possible in FTC, there are a few main downsides, especially as a beginner team to advanced software:

- The Lynx Hardware Manager used for FTC is blocking. This means without modifications, doing hardware calls on multiple threads can cause extreme performance issues in seemingly unexpected ways.
- Threading related bugs such as race conditions can be incredibly hard to debug!
- A command-based system can achieve very similar results!

Let's break it down- Commands

Every command is an individual action, that specifies 4 things about itself:

Start
An action that happens when the command is first added

End
An action that happens when the command ends, or stops ticking

Tick
An action that happens every single loop that the command is active

Is Complete
Every single tick, the command is evaluated to see if it is complete. If it is, execution stops.

You can think of them as miniature OpModes.

Commands

Robot

Subsystem
- Encapsulates the functionality of hardware

Lift Move Command
- Init: Sets the target position of the lift
- Tick: Nothing
- End: Nothing
- Is Complete: When the lift is within a certain tolerance of the target position

Lift
- Contains an amount of motors
- Manages the control of the motors to go to a certain height

Drivetrain
- Contains all drive motors.
- Exposes a "drive" method that takes a direction to drive in.

Command Chaining

Linear Series Command

```
fun deposit() = series(
        PrimeLiftCommand(),
        LiftMoverCommand(height = 9.2, tolerance = 0.5),
        DepositCommand(),
        LiftReturnCommand()
 )
```

Pre-Structuring

Driver Controlled Example
Initialize all hardware (Bulky piece of code!)

When button is pressed:
- Move lift with to a specific point.

Mecanum drive kinematics & movement

Autonomous Example
Initialize all hardware (Bulky piece of code!)

Mecanum drive kinematics & movement to follow path
Move lift to a specific point

Duplicated bulky code
This can be improved!

Post-Structuring

Driver Controlled Example

When button is pressed, run **Lift Move Command**

Run **Joystick Drive Command**

Autonomous Example

Run **Path Follower Command** with a path.

Run **Lift Move Command**

Lift Move

Joystick Drive

Path Follower

Robot

Lift

- Contains an amount of motors
- Manages the control of the motors to go to a certain height

Drivetrain

- Contains all drive motors.
- Exposes a "drive" method that takes a direction to drive in.

How can I structure my code like this?
Command-Based Implementations

A good option to consider, and our recommended one, is making your own!
It's always a good learning opportunity, and you can explore different structures much more.

Command-Based Implementations FTCLib

Pros:

Pretty good documentation

Established

Cons:

Shows its age- Doesn't follow Java conventions

Clunky at times, and the scheduler can be esoteric

```java
/**
 * A simple command that grabs a stone with the
 * {@link GripperSubsystem}.  Written explicitly for
 * pedagogical purposes. Actual code should inline a
 * command this simple with {@link
 * com.arcrobotics.ftclib.command.InstantCommand}.
 */
public class GrabStone extends CommandBase {

    // The subsystem the command runs on
    private final GripperSubsystem m_gripperSubsystem;

    public GrabStone(GripperSubsystem subsystem) {
        m_gripperSubsystem = subsystem;
        addRequirements(m_gripperSubsystem);
    }

    @Override
    public void initialize() {
        m_gripperSubsystem.grab();
    }

    @Override
    public boolean isFinished() {
        return true;
    }

}
```

```java
/**
 * An example command that uses an example subsystem.
 */
public class ExampleCommand extends CommandBase {
  @SuppressWarnings({"PMD.UnusedPrivateField", "PMD.SingularField"})
  private final ExampleSubsystem m_subsystem;

  /**
   * Creates a new ExampleCommand.
   *
   * @param subsystem The subsystem used by this command.
   */
  public ExampleCommand(ExampleSubsystem subsystem) {
    m_subsystem = subsystem;
    // Use addRequirements() here to declare subsystem dependencies.
    addRequirements(subsystem);
  }
}
```

Command-Based Implementations

nautilus

Work in progress- Not released yet.

Nautilus is a modular culmination of the work of our team over the past few years in FTC programming in Kotlin.

Command-Based Implementations

Turtle Studio

OpModes Worlds - Blue - Driver Control Stop

Status Connected

Inspector

- LakituCapper
- LakituDuck
- LakituLift
- LakituIntake
- JoystickDriveComponent
- CapTurretAdjusterComponent

Telemetry

movement vector:	Vector2(0.0, -0.0)
turning:	0.0
cap_yaw:	-1.0
cap_pitch:	-0.5
volts:	12.537
lift sensor distance:	5.257746568369858
left_lift_encoder:	0.0
right_lift_encoder:	0.0
fps:	25.0
circles:	

Graph

telemetry.duckGoal * 50 + 50

telemetry.fps

Logs

[TELEMETRY] - Component LakituLift: looped: time 4ms
[TELEMETRY] - Component LakituIntake: looped: time 0ms
[TELEMETRY] - Component CapTurret: looped: time 0ms
[TELEMETRY] [InstantCommand] took -1ms for activate instant
[TELEMETRY] [InstantCommand] took 0ms for activate instant

Info Status Telemetry Error Clear

Hardware

frontLeft
Power: 0.00
power: 0 min: -1 max: 1 encoder: 0

frontRight
Power: 0.00

Inspector

- S-Odometry
- Mecanum DT
- Lift
- GVF
  - kN 0.1
  - speed 0
  - reverse
  - invert

Status Disconnected

OpModes ReplOpMode Init

Hardware

retract1
Power: 0.40
Robot Control Manual
An OpMode must be running to use Hardware.
retract2
Power: 0.00
Robot Control Manual

Logs

[STATUS] OpMode `ReplOpMode` started!
[TELEMETRY] Original PID is P : 0.0 | I : 0.1 | D : 0.2
[TELEMETRY] Original speed is 0.75
[TELEMETRY] Original play is is false
[INFO] Initialized `GuidingVectorField`.
[STATUS] OpMode `ReplOpMode` stopped!

Info Status Telemetry Error Clear

Macros

setpos auto Run
+ Add Macro
An OpMode must be running to use Macros.

Repl

An OpMode must be running to use the REPL.

Command-Based Implementations

nautilus

```kotlin
/**
 * FTC #16461 Robot.
 */
class Oogway: Robot() {

    /* Drivetrain motors */
    val frontLeft =     motor("frontLeft").brake(true).reverse(true)
    val frontRight =    motor("frontRight").brake(true).reverse(true)
    val backLeft =      motor("backLeft").brake(true).reverse(true)
    val backRight =     motor("backRight").brake(true)

    /* Scoring mech motors */
    val intakeMotor = motor("intake")

    /* Deadwheels */
    val leftDeadwheel = frontLeft.encoder.reverse(true)
    val backDeadwheel = backLeft.encoder

    ...
```

```kotlin
await(series(
        idler { _, _ -> !arm.liftIsTransitioning },
        when(capstonePosition) {
            CapstonePosition.LEFT -> arm.highGoal()
            CapstonePosition.MIDDLE -> arm.midGoal()
            CapstonePosition.RIGHT -> arm.lowGoal()
            else -> arm.intake()
        }
))

await(delay(1.5))

await(follow(
        pos,
        pose(10.inches, 8.inches, 0.degrees),
        pose(43.inches, 43.inches, 0.degrees),
        speed = 0.35
))
```

Keep up to date with Nautilus!

nautilus.mcr.club

Main Takeaways

- Separate logic away from OpModes
- Don't use threading unless necessary (machine vision)
- Separate commands, subsystems, and the robot from OpModes.
- Experiment! No one code structure will work for everyone.

Contacts and Help

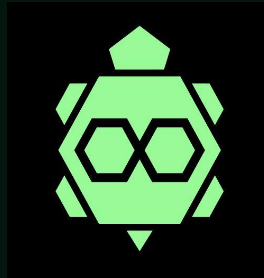
We are both from 16461, a team based in Southeast Charlotte, and are occasionally able to help in-person in the Charlotte Metro area.

We can be contacted with our emails at **asher@mcr.club** and **ryan@mcr.club**, please CC a coach on your communications.
We can be contacted on discord **@ashermyers** and **@ryanhcode**, preferably being pinged on the NCFTC or 16461 discord.

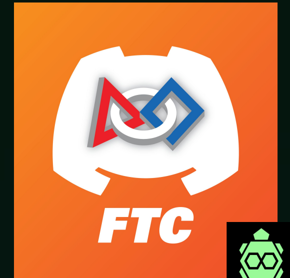
Teams can join our discord and gain access to a help channel at https://discord.gg/nEFb7X5BUR
We recommend teams join the NCFTC discord for help from other state teams at https://discord.gg/cEhWHYBmvU
We also recommend teams join the global FTC discord, partially moderated by our team, at https://discord.gg/first-tech-challenge

This presentation and all other 16461 kickoff presentations can be found on 16461's website at https://16461.mcr.club

https://discord.gg/cEhWHYBmvU



https://16461.mcr.club
https://discord.gg/nEFb7X5BUR



https://discord.gg/first-tech-challenge