**Behaviors from user actions and triggers**

How do we handle conflicting inputs???

**Need to update or verify these items**

Extend intake (Joy 2 button 1): Requires intake. Uses uptake but does not require it.

* Extend intake
* Turn on intake motor, transfer motor
* If there is no ball at the color sensor, then turns on the uptake motor and turns off the eject motor. It does this without requiring the uptake, so it could cause interfering behaviors.

Close Intake (Joy 2 button 2): Requires intake, uptake. Uses feeder ball sensor.

* Retract intake
* If 2 balls, then turn off intake motor, transfer motor, uptake motor
* If 0 or 1 balls, then run flush sequence, then turn off intake motor, transfer motor, uptake motor

Shoot sequence (RT): Requires uptake, feeder, shooter

* Why so many required subsystem????
* NOT interruptible

Sort ball at uptake (trigger when ball is at color sensor in teleop): Requires uptake, feeder. Forks intake (does not require intake).

* NOT interruptible
* Turn off feeder
* Wait 0.15 sec for color sensor to stabilize
* If wrong color
  + Eject ball – run eject motor until ball is not at color sensor (with min/max times)
  + Leave uptake motor running
* If right color
  + If a ball is not in feeder (this is first ball)
    - X-box rumble once
    - Set eject motor to run until ball is in feeder, then turn off eject motor
    - Leave uptake motor running
  + If there already is a ball in the feeder (this is 2nd ball)
    - X-box rumble twice
    - Fork to retract intake, turn off intake motor, turn off transfer motor
    - Turn off uptake motor

**Current behaviors are correct for these items**

Set shooter speed short shot (A): Requires shooter, [pi – no requirement]

Set shooter speed medium shot (B): Requires shooter, [pi – no requirement]

Set shooter speed long shot (Y): Requires shooter, [pi – no requirement]

Set shooter speed pit shot (Back): Requires shooter, [pi – no requirement]

Set shooter speed using vision (X): Requires shooter, [pi – no requirement]

* OK to interrupt. These sequences set the Falcon PID to the selected speed when initialized. Execute does nothing. The sequence naturally ends when the shooter is at the requested speed, but if interrupted then the Falcon PID will continue to control the motor to the selected speed.

Turn turret left manually (LB) (release to stop): Requires turret

Turn turret right manually (RB) (release to stop): Requires turret

* OK to interrupt. Note that releasing the button will stop the turret from moving (could interrupt automated turret movement).
* Potential improvement: Update TurretSetPercentOutput to have an option to keep running and set power to 0 when interrupted. Then change these buttons from whenPressed to whileHeld and removed the whenReleased. Behavior should be the same for these buttons. However, releasing LB/RB would no longer interrupt other sequences. Other options include decorators, such as .perpetually().

Turn turret 0 (D-pad up): Requires turret, [pi – no requirement]

Turn turret -45 (D-pad left): Requires turret, [pi – no requirement]

Turn turret +45 (D-pad right): Requires turret, [pi – no requirement]

* OK to interrupt, which will stop the motor.

Aim using vision (LT) (hold to track, release to stop): Requires turret, [pi – no requirement]