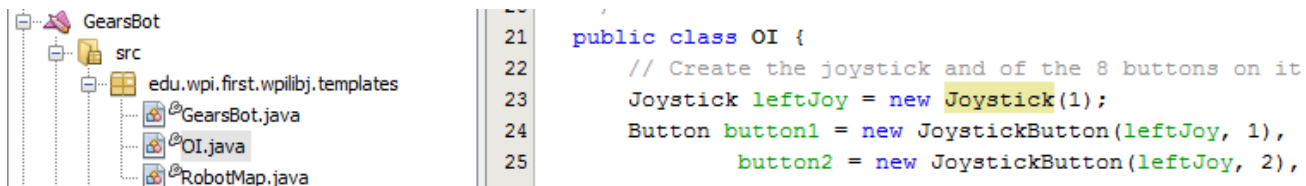


Running commands on Joystick input

Running commands on Joystick input

You can cause commands to run when joystick buttons are pressed, released, or continuously while the button is held down. This is extremely easy to do only requiring a few lines of code.

The OI Class



The command based template contains a class called OI, located in OI.java, where Operator Interface behaviors are typically defined. If you are using RobotBuilder this file can be found in the org.usfirst.frc####.NAME package

Create the Joystick object and JoystickButton objects

C++

```
OI::OI()
{
    joy = new Joystick(1);

    JoystickButton* button1 = new JoystickButton(joy, 1),
    button2 = new JoystickButton(joy, 2),
    button3 = new JoystickButton(joy, 3),
    button4 = new JoystickButton(joy, 4),
    button5 = new JoystickButton(joy, 5),
    button6 = new JoystickButton(joy, 6),
    button7 = new JoystickButton(joy, 7),
    button8 = new JoystickButton(joy, 8);
}
```

Running commands on Joystick input

Java

```
public class OI {  
    // Create the joystick and the 8 buttons on it  
    Joystick leftJoy = new Joystick(1);  
    Button button1 = new JoystickButton(leftJoy, 1),  
        button2 = new JoystickButton(leftJoy, 2),  
        button3 = new JoystickButton(leftJoy, 3),  
        button4 = new JoystickButton(leftJoy, 4),  
        button5 = new JoystickButton(leftJoy, 5),  
        button6 = new JoystickButton(leftJoy, 6),  
        button7 = new JoystickButton(leftJoy, 7),  
        button8 = new JoystickButton(leftJoy, 8);  
  
}
```

In this example there is a Joystick object connected as Joystick 1. Then 8 buttons are defined on that joystick to control various aspects of the robot. This is especially useful for testing although generating buttons on SmartDashboard is another alternative for testing commands.

Associate the buttons with commands

C++

```
button1->WhenPressed(new PrepareToGrab());  
button2->WhenPressed(new Grab());  
button3->WhenPressed(new DriveToDistance(0.11));  
button4->WhenPressed(new PlaceSoda());  
button6->WhenPressed(new DriveToDistance(0.2));  
button8->WhenPressed(new Stow());  
  
button7->WhenPressed(new SodaDelivery());
```

Running commands on Joystick input

Java

```
public OI() {  
    button1.whenPressed(new PrepareToGrab());  
    button2.whenPressed(new Grab());  
    button3.whenPressed(new DriveToDistance(0.11));  
    button4.whenPressed(new PlaceSoda());  
    button6.whenPressed(new DriveToDistance(0.2));  
    button8.whenPressed(new Stow());  
  
    button7.whenPressed(new SodaDelivery());  
}
```

In this example most of the joystick buttons from the previous code fragment are associated with commands. When the associated button is pressed the command is run. This is an excellent way to create a teleop program that has buttons to do particular actions.

Other options

In addition to the "whenPressed()" condition showcased above, there are a few other conditions you can use to link buttons to commands:

- Commands can run when a button is released by using whenReleased() instead of whenPressed().
- Commands can run continuously while the button is depressed by calling whileHeld().
- Commands can be toggled when a button is pressed using toggleWhenPressed().
- A command can be canceled when a button is pressed using cancelWhenPressed().

Additionally commands can be triggered by arbitrary conditions of your choosing by using the Trigger class instead of Button. Triggers (and Buttons) are usually polled every 20ms or whenever the scheduler is called.