Limiting Buttons by Time and Toggles

July 10, 2018

Introduction

Toggles and button “cooldowns” are both incredibly useful for ease of driving your robot. Button cooldowns prevent the driver from pressing the same button again for a certain amount of time. This feature is useful to prevent triggering a function too many times. Toggles are useful for functions that are required to be called continuously, but the driver does not want to hold the button down. Cooldowns are necessary to use toggles to prevent the toggle variable from switching to true then back to false with one press.

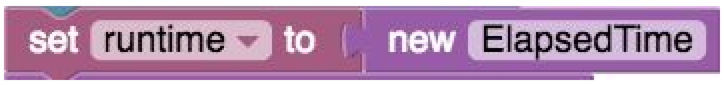
Table of Contents

Cooldowns

Cooldowns compare the current time to the most recent time a certain button was used. The times will be tracked using ElapsedTime, called upon using:

```
private ElapsedTime runtime = new ElapsedTime();
```

or



In both the java and Blocks example, I've created a variable, runtime, as a measure of the ElapsedTime. All blocks related to ElapsedTime are under Utilities > Time > ElapsedTime.

Then you can use runtime.milliseconds() or runtime.seconds() to get the amount of time the teleop has been running for. In Blocks, you can socket your runtime variable into the ElapsedTime.milliseconds block. The difference in time between the current time and the last use time is how the elapsed time since the last use of a key is determined.

```
runtime.milliseconds() - lastUseTime.a > 1000
```

The above statement returns true if a was used more than 1000 milliseconds ago, and false if a was used less than 1000 milliseconds ago. The following function returns a boolean based on the cooldown time of a key.

```
public boolean cdCheck(double key, int cdTime){
    return runtime.milliseconds() - key > cdTime;
}
```

In the main teleop loop you would use the following:

```
if(gamepad1.a && cdCheck(useMap1.a, 1000)){
    doStuffWhenPressAIfNotOnCooldown();
    useMap1.a = runtime.milliseconds();
}
```

Which checks if *gamepad1.a* is pressed, then checks if *gamepad1.a* has been used in the past 1000 milliseconds, then it runs the desired code. I also included resetting *useMap1.a* to *runtime.milliseconds()* (the current time) so that the most recent time *gamepad1.a* was used is set to the current time.

Another feature worth looking at is how *useMap1* is declared. In a separate java file called *useMap*, several variables are listed including *a*. By declaring *useMap1* at the beginning of the teleop (*useMap useMap1 = new useMap();*) we can access *useMap1.a*. We can do the same method to define *useMap2* (*useMap useMap2 = new useMap();*) and access *useMap2.a*. I've designated *useMap1* to contain all the use times for *gamepad1* and *useMap2* to contain all the use times for *gamepad2*. GitHub link to useMap.java.

For Blocks, each useTime variable and each toggle variable must be declared individually. This approach can also be used for java, but the useMap approach written above is more convenient.

Toggles

Toggles are booleans that act as a more advanced control than button pressing. Instead of holding down A, the driver can press A once and the action will be run continuously until A is pressed again. The simplified logic of a toggle changes the toggle variable (*toggleMap1.a*) to true if it's false and false if it's true whenever *gamepad1.a* is pressed.

Here's the simple code for a toggle.

```
public boolean toggle(boolean variable){
    if(variable == true){
        variable = false;
    }
    else if(variable == false){
        variable = true;
    }
    return variable;
}
```

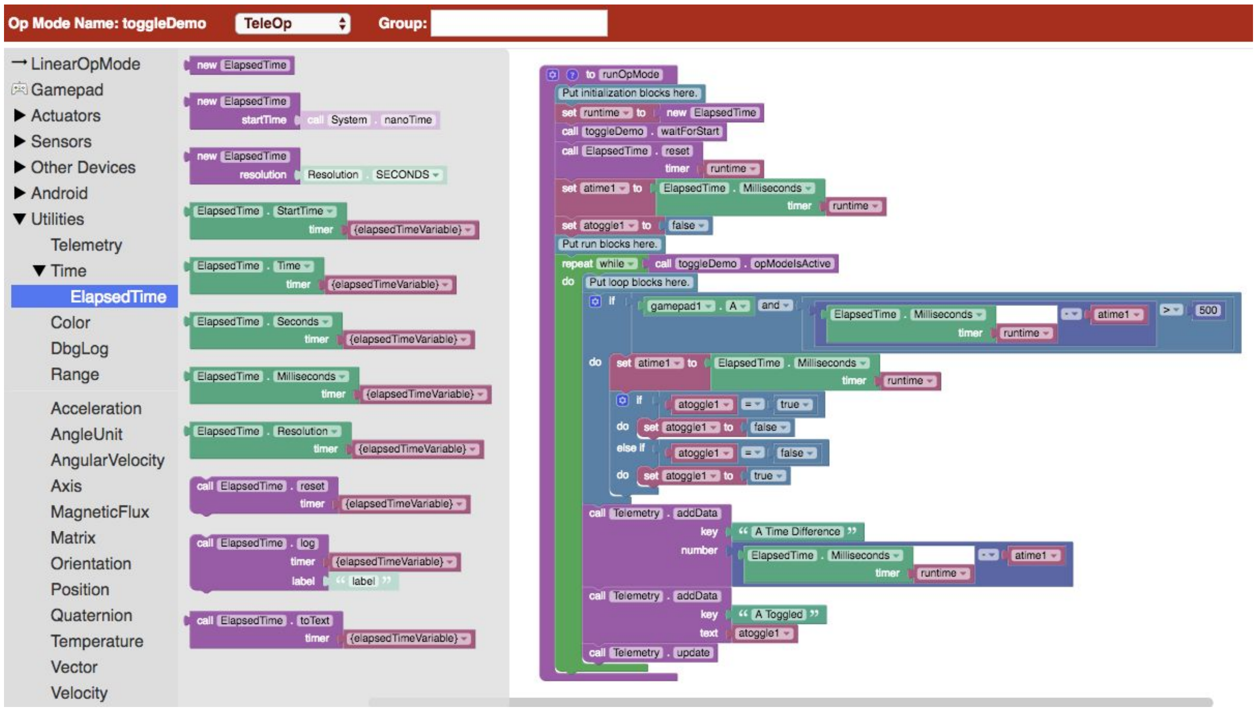
To use this function, set `toggleMap1.a = toggle(toggleMap1.a);`

The action of pressing *gamepad1.a* requires a cooldown to work. Without a cooldown, *toggleMap1.a* could switch from false to true and back to false even if you tap *gamepad1.a* briefly. We can put the code to toggle *toggleMap1.a* between true and false inside the cooldown check shown above.

Links and Example Code

Github link to java code teleop.
Github link to java useMap.
Github link to java toggleMap.
Github link to Blocks teleop.

```
package org.firstinspires.ftc.teamcode;

import com.qualcomm.robotcore.eventloop.opmode.LinearOpMode;
import com.qualcomm.robotcore.eventloop.opmode.TeleOp;
import com.qualcomm.robotcore.hardware.DcMotor;
import com.qualcomm.robotcore.util.ElapsedTime;

@TeleOp(name="keyDemo", group="Test")
public class keyDemo extends LinearOpMode {
    private ElapsedTime runtime = new ElapsedTime();
    //////////// NOTES \\\\\\\\\\\\
    /*
    Insert useMap1.a = runtime.milliseconds(); after every use of gamepad1.a to reset the cooldown.
    Use toggleMap1.a to access whether gamepad1.a is toggled or not.
    cdCheck(useMap1.a, 1000) returns true or false based on whether gamepad1.a has been used in the
last 1000 milliseconds.
    */
    toggleMap toggleMap1 = new toggleMap();
    useMap useMap1 = new useMap();

    toggleMap toggleMap2 = new toggleMap();
    useMap useMap2 = new useMap();
    @Override
    public void runOpMode() {

        while(!opModeIsActive()){}

        while(opModeIsActive()){
            updateKeys();
            telemetry.addData("a", toggleMap1.a + " " + (runtime.milliseconds() - useMap1.a));
            telemetry.addData("b", toggleMap1.b + " " + (runtime.milliseconds() - useMap1.b));
            telemetry.addData("b2", toggleMap2.b + " " + (runtime.milliseconds() - useMap2.b));
            telemetry.addData("right_stick_x_pos", toggleMap1.right_stick_x_pos + " " +
(runtime.milliseconds() - useMap1.right_stick_x_pos));
            telemetry.update();
        }
    }
    public void updateKeys(){
        if(gamepad1.a && cdCheck(useMap1.a, 1000)){
            toggleMap1.a = toggle(toggleMap1.a);
            useMap1.a = runtime.milliseconds();
        }
        if(gamepad1.b && cdCheck(useMap1.b, 500)){
            toggleMap1.b = toggle(toggleMap1.b);
            useMap1.b = runtime.milliseconds();
        }
        if(gamepad2.b && cdCheck(useMap2.b, 500)){
            toggleMap2.b = toggle(toggleMap2.b);
            useMap2.b = runtime.milliseconds();
        }
        if(gamepad1.right_stick_x > 0 && cdCheck(useMap1.right_stick_x_pos, 700)){
            toggleMap1.right_stick_x_pos = toggle(toggleMap1.right_stick_x_pos);
            useMap1.right_stick_x_pos = runtime.milliseconds();
        }
    }
    public boolean cdCheck(double key, int cdTime){
        return runtime.milliseconds() - key > cdTime;
    }
    public boolean toggle(boolean variable){
        if(variable == true){
            variable = false;
        }
        else if(variable == false){
            variable = true;
        }
        return variable;
    }
}
```



```
to runOpMode
  Put initialization blocks here.
  set runtime to new ElapsedTime
  call toggleDemo . waitForStart
  call ElapsedTime . reset
                     timer runtime
  set atime1 to ElapsedTime . Milliseconds
                              timer runtime
  set atoggle1 to false
  Put run blocks here.
  repeat while call toggleDemo . opModeIsActive
  do Put loop blocks here.
     if gamepad1 . A and ElapsedTime . Milliseconds - atime1 > 500
                                       timer runtime
     do set atime1 to ElapsedTime . Milliseconds
                                    timer runtime
        if atoggle1 = true
        do set atoggle1 to false
        else if atoggle1 = false
        do set atoggle1 to true
     call Telemetry . addData
                      key "A Time Difference"
                      number ElapsedTime . Milliseconds - atime1
                                           timer runtime
     call Telemetry . addData
                      key "A Toggled"
                      text atoggle1
     call Telemetry . update
```

Contact Information

Feel free to ask me anything about this guide or any other questions you have about robotics! Send emails to engineering@yale.edu or you can message me on Discord at nahn20#5234. I use Github to upload the example teleops in these guides. All Yale School of Engineering & Applied Science guides can be found here.