**Code 26NOV16 – Base FFTank Op Mode – Todd Ferguson**

**package** com.qualcomm.ftcrobotcontroller.opmodes;  
  
**import** com.qualcomm.robotcore.eventloop.opmode.OpMode;  
**import** com.qualcomm.robotcore.hardware.DcMotor;  
**import** com.qualcomm.robotcore.hardware.Servo;  
**import** com.qualcomm.robotcore.util.Range;  
  
*/\*\*  
 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 \* Opmode Name: FFTank.java  
 \* Author: FRC 503-FrogForce  
 \* Description: Demonstrates TeleOp tank drive mode using PushBot motor/server names  
 \*  
 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/***public class** FFTank **extends** OpMode {  
  
 */\*  
 \* Note: the configuration of the servos is such that  
 \* as the arm servo approaches 0, the arm position moves up (away from the floor).  
 \* Also, as the claw servo approaches 0, the claw opens up (drops the game element).  
 \*/  
 // TETRIX VALUES.* **final static double *ARM\_MIN\_RANGE*** = 0.20;  
 **final static double *ARM\_MAX\_RANGE*** = 0.90;  
 **final static double *CLAW\_MIN\_RANGE*** = 0.20;  
 **final static double *CLAW\_MAX\_RANGE*** = 0.7;  
  
 *// position of the hand servos.* **double handPosition**;  
  
 *// position of the tape measure servo* **double tapeposition**;  
  
 *// amount to change the arm servo position.* **double armDelta** = 0.1;  
  
 *// amount to change the claw servo position by* **double handDelta** = 0.05;  
  
 *// amount to change the tape servo position by* **double tapedelta** = 0.05;  
  
 DcMotor **motorRight**;  
 DcMotor **motorLeft**;  
 DcMotor **motorArm**;  
 DcMotor **motorTurret**;  
 Servo **leftHand**;  
 Servo **rightHand**;  
 Servo **tape**;  
  
  
  
 */\*\*  
 \* Constructor  
 \*/* **public** FFTank() {  
  
 }  
  
 */\*  
 \* Code to run when the op mode is first enabled goes here  
 \*  
 \* @see com.qualcomm.robotcore.eventloop.opmode.OpMode#start()  
 \*/* @Override  
 **public void** init() {  
 */\*  
 \* Use the hardwareMap to get the dc motors and servos by name. Note  
 \* that the names of the devices must match the names used when you  
 \* configured your robot and created the configuration file.  
 \*/* **motorRight** = **hardwareMap**.**dcMotor**.get(**"right\_drive"**);  
 **motorLeft** = **hardwareMap**.**dcMotor**.get(**"left\_drive"**);  
 **motorLeft**.setDirection(DcMotor.Direction.***REVERSE***);  
 **motorArm** = **hardwareMap**.**dcMotor**.get(**"left\_arm"**);  
 **motorTurret** = **hardwareMap**.**dcMotor**.get(**"right\_arm"**);  
  
 **leftHand** = **hardwareMap**.**servo**.get(**"left\_hand"**);  
 **rightHand** = **hardwareMap**.**servo**.get(**"right\_hand"**);  
 **tape** = **hardwareMap**.**servo**.get(**"tape"**);  
  
  
 *// assign the starting position of the left and right hand* **handPosition** = 0.5;  
 *// assign the starting position of the tape measure* **tapeposition** = 0.5;  
 }  
  
 */\*  
 \* This method will be called repeatedly in a loop  
 \*  
 \* @see com.qualcomm.robotcore.eventloop.opmode.OpMode#run()  
 \*/* @Override  
 **public void** loop() {  
  
 */\*  
 \* Gamepad 1  
 \*  
 \* Gamepad 1 controls the motors via the left stick and right sticks  
 \*  
 \* GamePad 2  
 \* it Controls the arm with the left joystick and the turret with right joystick  
 \* if controls the left hand x,b buttons  
 \*/  
  
 // tank drive  
 // note that if y equal -1 then joystick is pushed all of the way forward.* **float** left = -**gamepad1**.**left\_stick\_y**;  
 **float** right = -**gamepad1**.**right\_stick\_y**;  
  
 *// clip the right/left values so that the values never exceed +/- 1* right = Range.*clip*(right, -1, 1);  
 left = Range.*clip*(left, -1, 1);  
  
 *// scale the joystick value to make it easier to control  
 // the robot more precisely at slower speeds.* right = (**float**)scaleInput(right);  
 left = (**float**)scaleInput(left);  
  
 *// write the values to the motors* **motorRight**.setPower(right);  
 **motorLeft**.setPower(left);  
  
 *//Manage the ARM motor* **float** arm\_left\_stick\_y = -**gamepad2**.**left\_stick\_y**;  
 **float** arm\_power = (**float**)scaleInput(arm\_left\_stick\_y);  
 **motorArm**.setPower(arm\_power);  
  
 *//Manage the Turret motor* **float** arm\_right\_stick\_y = -**gamepad2**.**right\_stick\_y**;  
 **float** turret\_power = (**float**)scaleInput(arm\_right\_stick\_y);  
 **motorTurret**.setPower(turret\_power);  
  
 *// Manage Tape Servo Motor  
 // if X button it pressed increase tape position  
 // if B button is pressed decrease tape position* **if** (**gamepad1**.**x**)  
 {  
 **tapeposition** = **tape**.getPosition() + **tapedelta**;  
 }  
 **else if** (**gamepad1**.**b**)  
 {  
 **tapeposition** = **tape**.getPosition() - **tapedelta**;  
 }  
 *// Added this to code to stop continuous servo motor with button Y by setting to value of 0.5* **else if** (**gamepad1**.**y**)  
 {  
 **tapeposition** = 0.5;  
 }  
  
  
  
 *// Manage Intake Servo Motors  
 // if X button it pressed increase hand position  
 // if B button is pressed decrease hand position* **if** (**gamepad2**.**x**)  
 {  
 **handPosition** = **leftHand**.getPosition() + **handDelta**;  
 }  
 **else if** (**gamepad2**.**b**)  
 {  
 **handPosition** = **leftHand**.getPosition() - **handDelta**;  
 }  
 *// Added this to code to stop continuous servo motor with button Y by setting to value of 0.5* **else if** (**gamepad2**.**y**)  
 {  
 **handPosition** = 0.5;  
 }  
  
  
 *//Set hand position  
 //Ensure the values are legal* **double** l\_position = Range.*clip* (  
 **handPosition** , Servo.***MIN\_POSITION*** , Servo.***MAX\_POSITION*** );  
 *//Set hand positions* **leftHand**.setPosition(l\_position);  
 **rightHand**.setPosition(1.0 - l\_position);  
  
  
 */\*  
 \* Send telemetry data back to driver station. Note that if we are using  
 \* a legacy NXT-compatible motor controller, then the getPower() method  
 \* will return a null value. The legacy NXT-compatible motor controllers  
 \* are currently write only.  
 \*/* **telemetry**.addData(**"Text"**, **"\*\*\* Robot Data\*\*\*"**);  
 *//telemetry.addData("arm", "arm: " + String.format("%.2f", arm\_Power));* **telemetry**.addData(**"hand"**, **"hand: "** + String.*format*(**"%.2f"**, **handPosition**));  
 **telemetry**.addData(**"left tgt pwr"**, **"left pwr: "** + String.*format*(**"%.2f"**, left));  
 **telemetry**.addData(**"right tgt pwr"**, **"right pwr: "** + String.*format*(**"%.2f"**, right));  
 }  
  
 */\*  
 \* Code to run when the op mode is first disabled goes here  
 \*  
 \* @see com.qualcomm.robotcore.eventloop.opmode.OpMode#stop()  
 \*/* @Override  
 **public void** stop() {  
  
 }  
  
 */\*  
 \* This method scales the joystick input so for low joystick values, the  
 \* scaled value is less than linear. This is to make it easier to drive  
 \* the robot more precisely at slower speeds.  
 \*/* **double** scaleInput(**double** dVal) {  
 **double**[] scaleArray = { 0.0, 0.05, 0.09, 0.10, 0.12, 0.15, 0.18, 0.24,  
 0.30, 0.36, 0.43, 0.50, 0.60, 0.72, 0.85, 1.00, 1.00 };  
  
 *// get the corresponding index for the scaleInput array.* **int** index = (**int**) (dVal \* 16.0);  
  
 *// index should be positive.* **if** (index < 0) {  
 index = -index;  
 }  
  
 *// index cannot exceed size of array minus 1.* **if** (index > 16) {  
 index = 16;  
 }  
  
 *// get value from the array.* **double** dScale = 0.0;  
 **if** (dVal < 0) {  
 dScale = -scaleArray[index];  
 } **else** {  
 dScale = scaleArray[index];  
 }  
  
 *// return scaled value.* **return** dScale;  
 }  
}

Changed to “else” statement only so that while button is pressed the servo turns else the position is set to 0.5 which stops the continuous servo.

**else if** (**gamepad2**.**y**)  
 {  
 **handPosition** = 0.5;

Needed Logitech F310 joystick assignments to program rest of robot.

Found on the ponytailposey website.

The following is a list of all buttons, and there names in the code. Note that each name must be prefixed with gamepad1 (ex: gamepad1.x)

Left joystick: left\_stick\_y (vertical) and left\_stick\_x (horizontal)

Right joystick: right\_stick\_y (vertical) and right\_stick\_x (horizontal)

For the joysticks, the value ranges between -1 (fully up) and 1 (fully down)

X button: x

A button: a

B button: b

Y button: y

DPad up: dpad\_up

DPad right: dpad\_right

DPad down: dpad\_down

DPad left: dpad\_left

Start: start

Mode: mode

Back: back

Right trigger: right\_trigger

Left trigger: left\_trigger

For the triggers, the values range from 0 (not pressed) to 1 (pressed).

Right bumper: right\_bumper

Left bumber: left\_bumper

**26NOV15 Rev2**

**package** com.qualcomm.ftcrobotcontroller.opmodes;  
  
**import** com.qualcomm.robotcore.eventloop.opmode.OpMode;  
**import** com.qualcomm.robotcore.hardware.DcMotor;  
**import** com.qualcomm.robotcore.hardware.Servo;  
**import** com.qualcomm.robotcore.util.Range;  
  
*/\*\*  
 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 \* Opmode Name: FFTank.java  
 \* Author: FRC 503-FrogForce  
 \* Description: Demonstrates TeleOp tank drive mode using PushBot motor/server names  
 \*  
 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/***public class** FFTank **extends** OpMode {  
  
 */\*  
 \* Note: the configuration of the servos is such that  
 \* as the arm servo approaches 0, the arm position moves up (away from the floor).  
 \* Also, as the claw servo approaches 0, the claw opens up (drops the game element).  
 \*/  
 // TETRIX VALUES.* **final static double *ARM\_MIN\_RANGE*** = 0.20;  
 **final static double *ARM\_MAX\_RANGE*** = 0.90;  
 **final static double *CLAW\_MIN\_RANGE*** = 0.20;  
 **final static double *CLAW\_MAX\_RANGE*** = 0.7;  
  
 *// position of the hand servos.* **double handPosition**;  
  
 *// position of the tape measure servo* **double tapeposition**;  
  
 *// amount to change the arm servo position.* **double armDelta** = 0.1;  
  
 *// amount to change the claw servo position by* **double handDelta** = 0.05;  
  
 *// amount to change the tape servo position by* **double tapedelta** = 0.05;  
  
 DcMotor **motorRight**;  
 DcMotor **motorLeft**;  
 DcMotor **motorArm**;  
 DcMotor **motorTurret**;  
 DcMotor **winch1**;  
 DcMotor **intake1**;  
 Servo **leftHand**;  
 Servo **rightHand**;  
 Servo **tape**;  
  
  
  
 */\*\*  
 \* Constructor  
 \*/* **public** FFTank() {  
  
 }  
  
 */\*  
 \* Code to run when the op mode is first enabled goes here  
 \*  
 \* @see com.qualcomm.robotcore.eventloop.opmode.OpMode#start()  
 \*/* @Override  
 **public void** init() {  
 */\*  
 \* Use the hardwareMap to get the dc motors and servos by name. Note  
 \* that the names of the devices must match the names used when you  
 \* configured your robot and created the configuration file.  
 \*/* **motorRight** = **hardwareMap**.**dcMotor**.get(**"right\_drive"**);  
 **motorLeft** = **hardwareMap**.**dcMotor**.get(**"left\_drive"**);  
 **motorLeft**.setDirection(DcMotor.Direction.***REVERSE***);  
 **motorArm** = **hardwareMap**.**dcMotor**.get(**"left\_arm"**);  
 **motorTurret** = **hardwareMap**.**dcMotor**.get(**"right\_arm"**);  
  
 **winch1** = **hardwareMap**.**dcMotor**.get(**"winch"**);  
 **intake1** = **hardwareMap**.**dcMotor**.get(**"intake"**);  
  
 **leftHand** = **hardwareMap**.**servo**.get(**"left\_hand"**);  
 **rightHand** = **hardwareMap**.**servo**.get(**"right\_hand"**);  
 **tape** = **hardwareMap**.**servo**.get(**"tape"**);  
  
  
 *// assign the starting position of the left and right hand* **handPosition** = 0.5;  
 *// assign the starting position of the tape measure* **tapeposition** = 0.5;  
 }  
  
 */\*  
 \* This method will be called repeatedly in a loop  
 \*  
 \* @see com.qualcomm.robotcore.eventloop.opmode.OpMode#run()  
 \*/* @Override  
 **public void** loop() {  
  
 */\*  
 \* Gamepad 1  
 \*  
 \* Gamepad 1 controls the motors via the left stick and right sticks  
 \*  
 \* GamePad 2  
 \* it Controls the arm with the left joystick and the turret with right joystick  
 \* if controls the left hand x,b buttons  
 \*/  
  
 // tank drive  
 // note that if y equal -1 then joystick is pushed all of the way forward.* **float** left = -**gamepad1**.**left\_stick\_y**;  
 **float** right = -**gamepad1**.**right\_stick\_y**;  
  
 *// clip the right/left values so that the values never exceed +/- 1* right = Range.*clip*(right, -1, 1);  
 left = Range.*clip*(left, -1, 1);  
  
 *// scale the joystick value to make it easier to control  
 // the robot more precisely at slower speeds.* right = (**float**)scaleInput(right);  
 left = (**float**)scaleInput(left);  
  
 *// write the values to the motors* **motorRight**.setPower(right);  
 **motorLeft**.setPower(left);  
  
 *//Manage the ARM motor* **float** arm\_left\_stick\_y = -**gamepad2**.**left\_stick\_y**;  
 **float** arm\_power = (**float**)scaleInput(arm\_left\_stick\_y);  
 **motorArm**.setPower(arm\_power);  
  
 *//Manage the Turret motor* **float** arm\_right\_stick\_y = -**gamepad2**.**right\_stick\_y**;  
 **float** turret\_power = (**float**)scaleInput(arm\_right\_stick\_y);  
 **motorTurret**.setPower(turret\_power);  
  
 *// Manage Tape Servo Motor  
 // if X button it pressed increase tape position  
 // if B button is pressed decrease tape position* **if** (**gamepad1**.**x**)  
 {  
 **tapeposition** = **tape**.getPosition() + **tapedelta**;  
 }  
 **else if** (**gamepad1**.**b**)  
 {  
 **tapeposition** = **tape**.getPosition() - **tapedelta**;  
 }  
 *// Added this to code to stop continuous servo motor with button Y by setting to value of 0.5* **else if** (**gamepad1**.**y**)  
 {  
 **tapeposition** = 0.5;  
 }  
  
  
  
 *// Manage Tape Servo Motors  
 // if X button it pressed increase hand position  
 // if B button is pressed decrease hand position* **if** (**gamepad1**.**a**)  
 {  
 **handPosition** = **leftHand**.getPosition() + **handDelta**;  
 }  
 **else if** (**gamepad1**.**y**)  
 {  
 **handPosition** = **leftHand**.getPosition() - **handDelta**;  
 }  
 *// Added this to code to stop continuous servo motor with button Y by setting to value of 0.5* **else** {  
 **handPosition** = 0.5;  
 }  
  
  
 *// Manage Intake Motor  
 // if X button it pressed intake objects into bucket  
 // if B button is pressed eject objects from bucket* **if** (**gamepad2**.**right\_bumper**)  
 {  
 **intake1**.setPower(1);  
 }  
 **else if** (**gamepad2**.**left\_bumper**)  
 {  
 **intake1**.setPower(-1);  
 }  
 *// Added this to code to stop continuous servo motor with button Y by setting to value of 0.5* **else** {  
 **intake1**.setPower(0);  
 }  
  
 *// Manage Winch Motor  
 // if X button it pressed intake objects into bucket  
 // if B button is pressed eject objects from bucket* **if** (**gamepad1**.**right\_bumper**)  
 {  
 **winch1**.setPower(1);  
 }  
 **else if** (**gamepad1**.**left\_bumper**)  
 {  
 **winch1**.setPower(-1);  
 }  
 *// Added this to code to stop continuous servo motor with button Y by setting to value of 0.5* **else** {  
 **winch1**.setPower(0);  
 }  
 *//Set hand position  
 //Ensure the values are legal* **double** l\_position = Range.*clip* (  
 **handPosition** , Servo.***MIN\_POSITION*** , Servo.***MAX\_POSITION*** );  
 *//Set hand positions* **leftHand**.setPosition(l\_position);  
 **rightHand**.setPosition(1.0 - l\_position);  
  
  
 */\*  
 \* Send telemetry data back to driver station. Note that if we are using  
 \* a legacy NXT-compatible motor controller, then the getPower() method  
 \* will return a null value. The legacy NXT-compatible motor controllers  
 \* are currently write only.  
 \*/* **telemetry**.addData(**"Text"**, **"\*\*\* Robot Data\*\*\*"**);  
 *//telemetry.addData("arm", "arm: " + String.format("%.2f", arm\_Power));* **telemetry**.addData(**"hand"**, **"hand: "** + String.*format*(**"%.2f"**, **handPosition**));  
 **telemetry**.addData(**"left tgt pwr"**, **"left pwr: "** + String.*format*(**"%.2f"**, left));  
 **telemetry**.addData(**"right tgt pwr"**, **"right pwr: "** + String.*format*(**"%.2f"**, right));  
 }  
  
 */\*  
 \* Code to run when the op mode is first disabled goes here  
 \*  
 \* @see com.qualcomm.robotcore.eventloop.opmode.OpMode#stop()  
 \*/* @Override  
 **public void** stop() {  
  
 }  
  
 */\*  
 \* This method scales the joystick input so for low joystick values, the  
 \* scaled value is less than linear. This is to make it easier to drive  
 \* the robot more precisely at slower speeds.  
 \*/* **double** scaleInput(**double** dVal) {  
 **double**[] scaleArray = { 0.0, 0.05, 0.09, 0.10, 0.12, 0.15, 0.18, 0.24,  
 0.30, 0.36, 0.43, 0.50, 0.60, 0.72, 0.85, 1.00, 1.00 };  
  
 *// get the corresponding index for the scaleInput array.* **int** index = (**int**) (dVal \* 16.0);  
  
 *// index should be positive.* **if** (index < 0) {  
 index = -index;  
 }  
  
 *// index cannot exceed size of array minus 1.* **if** (index > 16) {  
 index = 16;  
 }  
  
 *// get value from the array.* **double** dScale = 0.0;  
 **if** (dVal < 0) {  
 dScale = -scaleArray[index];  
 } **else** {  
 dScale = scaleArray[index];  
 }  
  
 *// return scaled value.* **return** dScale;  
 }  
}