**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.hardware.HardwareMap;**

**import com.qualcomm.robotcore.util.ElapsedTime;**

**import org.firstinspires.ftc.robotcore.external.Func;**

**import org.firstinspires.ftc.robotcore.external.navigation.AngleUnit;**

**import org.firstinspires.ftc.robotcore.external.navigation.AxesOrder;**

**import org.firstinspires.ftc.robotcore.external.navigation.AxesReference;**

**import org.firstinspires.ftc.robotcore.external.navigation.Position;**

**import org.firstinspires.ftc.robotcore.external.navigation.Velocity;**

**import java.util.Locale;**

***/\*\****

***\* Created by mandy.peake and helen.watson on 8/30/2019.***

***\*/***

**@TeleOp (name= "TeleOpRoverRuckus", group= "Linear Opmode")**

**public class TeleOpRoverRuckus extends LinearOpMode {**

**HardwareRoverRuckus Rover = new HardwareRoverRuckus();**

**ElapsedTime runtime = new ElapsedTime();**

**@Override**

**public void runOpMode() {**

**telemetry.addData("init pressed", "about to initialize");**

**telemetry.update();**

**Rover.initializeRobot(hardwareMap);**

**// Rover.rightCollector.setPosition(1);**

**// Rover.leftCollector.setPosition(0);**

**telemetry.addData("Status", "Ready to run");**

**telemetry.update();**

**composeTelemetry();**

**waitForStart();**

**// Rover.imu.startAccelerationIntegration(new Position(), new Velocity(), 1000);**

**while (opModeIsActive()) {**

**// telemetry.addData("armExtend", Rover.armExtend.getCurrentPosition());**

**// telemetry.addData("Left Arm Encoder", Rover.armLeft.getCurrentPosition());**

**// telemetry.addData("Right Arm Encoder", Rover.armRight.getCurrentPosition());**

**// telemetry.update();**

**double fwdBack = -gamepad1.right\_stick\_y;**

**double strafe = -gamepad1.right\_stick\_x;**

**double turn = -gamepad1.left\_stick\_x;**

**if (gamepad1.start) { // drive robot at slower speed for fine adjustments while carrying gold**

**Rover.leftFront.setPower((fwdBack + 1.5\*strafe - turn) \* .25);**

**Rover.leftBack.setPower((fwdBack - 1.5\*strafe - turn) \* .25);**

**Rover.rightFront.setPower((-fwdBack + 1.5\*strafe - turn) \* .25);**

**Rover.rightBack.setPower((-fwdBack - 1.5\*strafe - turn) \* .25);**

**} else { // drive robot normally at full speed**

**Rover.leftFront.setPower((fwdBack + strafe - turn));**

**Rover.leftBack.setPower((fwdBack - strafe - turn));**

**Rover.rightFront.setPower((-fwdBack + strafe - turn));**

**Rover.rightBack.setPower((-fwdBack - strafe - turn));**

**}**

**if (gamepad2.left\_bumper) {**

**Rover.liftL.setMode(DcMotor.RunMode.*STOP\_AND\_RESET\_ENCODER*);**

**Rover.liftL.setMode(DcMotor.RunMode.*RUN\_TO\_POSITION*);**

**Rover.liftL.setTargetPosition(100);**

**Rover.liftL.setPower(.5);**

**} else if (gamepad2.left\_trigger > 0.5) {**

**Rover.liftL.setMode(DcMotor.RunMode.*STOP\_AND\_RESET\_ENCODER*);**

**Rover.liftL.setMode(DcMotor.RunMode.*RUN\_TO\_POSITION*);**

**Rover.liftL.setTargetPosition(-100);**

**Rover.liftL.setPower(-.5);**

**} else {**

**Rover.liftL.setPower(0);**

**}**

**if (gamepad2.right\_bumper) {**

**Rover.liftU.setMode(DcMotor.RunMode.*STOP\_AND\_RESET\_ENCODER*);**

**Rover.liftU.setMode(DcMotor.RunMode.*RUN\_TO\_POSITION*);**

**Rover.liftU.setTargetPosition(100);**

**Rover.liftU.setPower(.5);**

**} else if (gamepad2.right\_trigger > 0.5) {**

**Rover.liftU.setMode(DcMotor.RunMode.*STOP\_AND\_RESET\_ENCODER*);**

**Rover.liftU.setMode(DcMotor.RunMode.*RUN\_TO\_POSITION*);**

**Rover.liftU.setTargetPosition(-100);**

**Rover.liftU.setPower(-.5);**

**} else {**

**Rover.liftU.setPower(0);**

**}**

**//ball collector thing**

**/\* if (gamepad1.y) {**

**Rover.leftCollector.setPosition(.7);**

**Rover.rightCollector.setPosition(.3);**

**} else if (gamepad1.a) {**

**Rover.leftCollector.setPosition(0);**

**Rover.rightCollector.setPosition(1);**

**}**

**\*/**

**/\* if (gamepad1.x){**

**Rover.rotateSweeper(1, 0);**

**} else if (gamepad1.b){**

**Rover.rotateSweeper(.5, .5);**

**}**

**\*/**

**//Ball Flap**

**/\***

**if (gamepad2.x) {**

**Rover.rotateFlap(1);**

**} else if (gamepad2.a){**

**Rover.rotateFlap(.5);**

**} else if (gamepad2.b){**

**Rover.rotateFlap(0);**

**}**

**\*/**

**/\***

**//Arm Movements**

**if (gamepad2.y) {**

**Rover.liftArmEncoder(-.5, -500, 1);**

**} else {**

**Rover.liftArm(gamepad2.right\_stick\_y \* .6 + gamepad2.left\_stick\_y);**

**}**

**\*/**

**//Arm Extending**

**/\***

**if (gamepad2.dpad\_up) {**

**Rover.armExtend.setMode(DcMotor.RunMode.RUN\_USING\_ENCODER);**

**Rover.armExtend.setPower(.5);**

**} else if (gamepad2.dpad\_down) {**

**Rover.armExtend.setMode(DcMotor.RunMode.RUN\_USING\_ENCODER);**

**Rover.armExtend.setPower(-.5);**

**} else {**

**Rover.armExtend.setPower(0);**

**}**

**\*/**

**}**

**}**

**//----------------------------------------------------------------------------------------------**

**// Telemetry Configuration**

**//----------------------------------------------------------------------------------------------**

**public void composeTelemetry() {**

**// At the beginning of each telemetry update, grab a bunch of data**

**// from the IMU that we will then display in separate lines.**

**telemetry.addAction(new Runnable() {**

**@Override**

**public void run() {**

**// Acquiring the angles is relatively expensive; we don't want**

**// to do that in each of the three items that need that info, as that's**

**// three times the necessary expense.**

**// Rover.angles = Rover.imu.getAngularOrientation(AxesReference.INTRINSIC, AxesOrder.ZYX, AngleUnit.DEGREES);**

**// Rover.gravity = Rover.imu.getGravity();**

**}**

**});**

**telemetry.addLine()**

**.addData("status", new Func<String>() {**

**@Override**

**public String value() {**

**return "";**

**// return Rover.imu.getSystemStatus().toShortString();**

**}**

**})**

**.addData("calib", new Func<String>() {**

**@Override**

**public String value() {**

**return "";**

**// return Rover.imu.getCalibrationStatus().toString();**

**}**

**});**

**telemetry.addLine()**

**.addData("heading", new Func<String>() {**

**@Override**

**public String value() {**

**return "";**

**// return formatAngle(Rover.angles.angleUnit, Rover.angles.firstAngle);**

**}**

**})**

**.addData("roll", new Func<String>() {**

**@Override**

**public String value() {**

**return "";**

**// return formatAngle(Rover.angles.angleUnit, Rover.angles.secondAngle);**

**}**

**})**

**.addData("pitch", new Func<String>() {**

**@Override**

**public String value() {**

**return "";**

**//return formatAngle(Rover.angles.angleUnit, Rover.angles.thirdAngle);**

**}**

**});**

**telemetry.addLine()**

**.addData("grvty", new Func<String>() {**

**@Override**

**public String value() {**

**return "";**

**//return Rover.gravity.toString();**

**}**

**})**

**.addData("mag", new Func<String>() {**

**@Override**

**public String value() {**

**return String.*format*(Locale.*getDefault*(), "%.3f"//,**

**// Math.sqrt(Rover.gravity.xAccel \* Rover.gravity.xAccel**

**// + Rover.gravity.yAccel \* Rover.gravity.yAccel**

**// + Rover.gravity.zAccel \* Rover.gravity.zAccel)**

**);**

**}**

**});**

**}**

**//----------------------------------------------------------------------------------------------**

**// Formatting**

**//----------------------------------------------------------------------------------------------**

**String formatAngle(AngleUnit angleUnit, double angle) {**

**return formatDegrees(AngleUnit.*DEGREES*.fromUnit(angleUnit, angle));**

**}**

**String formatDegrees(double degrees) {**

**return String.*format*(Locale.*getDefault*(), "%.1f", AngleUnit.*DEGREES*.normalize(degrees));**

**}**

**}**