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

import com.qualcomm.robotcore.eventloop.opmode.Autonomous;
import com.qualcomm.robotcore.eventloop.opmode.Disabled;
import org.firstinspires.ftc.robotcore.external.ClassFactory;
import
org.firstinspires.ftc.robotcore.external.navigation.VuMarkInstanceId;
import
org.firstinspires.ftc.robotcore.external.navigation.RelicRecoveryVuMar
k;
import
org.firstinspires.ftc.robotcore.external.navigation.VuforiaTrackableDe
faultListener;
import org.firstinspires.ftc.robotcore.external.matrices.VectorF;
import
org.firstinspires.ftc.robotcore.external.navigation.Orientation;
import
org.firstinspires.ftc.robotcore.external.navigation.AxesReference;
import org.firstinspires.ftc.robotcore.external.navigation.AxesOrder;
import org.firstinspires.ftc.robotcore.external.navigation.AngleUnit;
import
org.firstinspires.ftc.robotcore.external.navigation.VuforiaTrackables;
import
org.firstinspires.ftc.robotcore.external.navigation.VuforiaTrackable;
import org.firstinspires.ftc.robotcore.external.matrices.OpenGLMatrix;
import
org.firstinspires.ftc.robotcore.external.navigation.VuforiaLocalizer;
import com.qualcomm.robotcore.eventloop.opmode.OpMode;
import com.qualcomm.robotcore.hardware.ColorSensor;
import com.qualcomm.robotcore.hardware.DistanceSensor;
import com.qualcomm.robotcore.hardware.CRServo;
import com.qualcomm.robotcore.hardware.Servo;
import com.qualcomm.robotcore.hardware.DcMotor;
import com.qualcomm.robotcore.eventloop.opmode.LinearOpMode;
import android.app.Activity;
import android.graphics.Color;
import android.view.View;

@Autonomous (name = "Blue Back", group = "Blue")
//@Disabled
public class BlueBack extends LinearOpMode
{

    DcMotor leftWheel = null;

    DcMotor rightWheel = null;

    DcMotor pulley = null;

    Servo colorArm;

    Servo leftGripper;

    Servo rightGripper;

    ColorSensor sensorColor;
```

```
    DistanceSensor sensorDistance;

    ControlClass controlClass;

    EncoderClass encoder;

    boolean isVuforiaScanned = false;

    public static final String TAG = "Vuforia VuMark Sample";

    OpenGLMatrix lastLocation = null;

    VuforiaLocalizer vuforia;

    @Override
    public void runOpMode () throws InterruptedException
    {

        leftWheel = hardwareMap.get(DcMotor.class, "leftWheel");


//leftWheel.setZeroPowerBehavior(DcMotor.ZeroPowerBehavior.BRAKE);

        rightWheel = hardwareMap.get(DcMotor.class, "rightWheel");



//rightWheel.setZeroPowerBehavior(DcMotor.ZeroPowerBehavior.BRAKE);

        pulley = hardwareMap.get(DcMotor.class, "pulley");

        pulley.setZeroPowerBehavior(DcMotor.ZeroPowerBehavior.BRAKE);

        colorArm = hardwareMap.get(Servo.class, "colorArm");

        leftGripper = hardwareMap.get(Servo.class, "leftGripper");

        rightGripper = hardwareMap.get(Servo.class, "rightGripper");

        sensorDistance = hardwareMap.get(DistanceSensor.class,
"colorSensor");

        sensorColor = hardwareMap.get(ColorSensor.class,
"colorSensor");

        controlClass = new ControlClass (leftWheel, rightWheel,
pulley, colorArm, leftGripper, rightGripper, false, false);

        encoder = new EncoderClass (leftWheel, rightWheel);

        controlClass.stopDriving ();

        controlClass.stopPulley ();

        controlClass.liftArm ();
```

```
        boolean isVuforiaScaned = false;

        float hsvValues[] = {0F, 0F, 0F};

        final float values[] = hsvValues;

        final double SCALE_FACTOR = 255;

        int relativeLayoutId =
hardwareMap.appContext.getResources().getIdentifier("RelativeLayout",
"id", hardwareMap.appContext.getPackageName());
        final View relativeLayout = ((Activity)
hardwareMap.appContext).findViewById(relativeLayoutId);

        int cameraMonitorViewId =
hardwareMap.appContext.getResources().getIdentifier("cameraMonitorView
Id", "id", hardwareMap.appContext.getPackageName());
            VuforiaLocalizer.Parameters parameters = new
VuforiaLocalizer.Parameters(cameraMonitorViewId);

            parameters.vuforiaLicenseKey =
"AeTp02//////AAAAGRb39XgIz0/9sVz+xN6dPHpdS3/6EixBCKYbNr9Hw/vPNumcWEdd9
x4Hbk5rSeIGWS5+Of5euIDm3rKmE7GhbnVC4inOw+R5+sFSI63Qd/JVk+mg5bgXatQF2n7
3NPzKLlqMfQ6JPAEiWYzOrz5C0Sn3Cv9y3hejMCbf4eg9BmvHJogCq78iEjgShpjdWP+8g
Z5IsiXPTOLsE9pX3Zz5FV8HMDSFKpF/q0SS5IA0WyInTteYTu2Dx9glOrlIkTjrnzUJtPg
YqJ6+HJbXcnOxXyKnkT4zmxY/R/RcWF9cAx0gmSC9YmJGNxIc3rth9xv1X/UjpN7kFMbgH
aGAL/QmpOyk8cffciuqbfiViagBsucS";

            parameters.cameraDirection =
VuforiaLocalizer.CameraDirection.BACK;
            this.vuforia =
ClassFactory.createVuforiaLocalizer(parameters);

            VuforiaTrackables relicTrackables =
this.vuforia.loadTrackablesFromAsset("RelicVuMark");
            VuforiaTrackable relicTemplate = relicTrackables.get(0);
            relicTemplate.setName("relicVuMarkTemplate");

        waitForStart ();

        encoder.reset();

        relicTrackables.activate();

        controlClass.closeGrippers ();

        controlClass.movePulleyUp (1);

        controlClass.dropArm ();

        while (opModeIsActive())
        {

            Color.RGBToHSV((int) (sensorColor.red() * SCALE_FACTOR),
                    (int) (sensorColor.green() * SCALE_FACTOR),
                    (int) (sensorColor.blue() * SCALE_FACTOR),
```

```
                    hsvValues);

            RelicRecoveryVuMark vuMark =
RelicRecoveryVuMark.from(relicTemplate);
            if (vuMark != RelicRecoveryVuMark.UNKNOWN) {

                telemetry.addData("VuMark", "%s visible", vuMark);
                telemetry.update();

                OpenGLMatrix pose =
((VuforiaTrackableDefaultListener)relicTemplate.getListener()).getPose
();


                if (pose != null) {
                    VectorF trans = pose.getTranslation();
                    Orientation rot = Orientation.getOrientation(pose,
AxesReference.EXTRINSIC, AxesOrder.XYZ, AngleUnit.DEGREES);

                    double tX = trans.get(0);
                    double tY = trans.get(1);
                    double tZ = trans.get(2);


                    double rX = rot.firstAngle;
                    double rY = rot.secondAngle;
                    double rZ = rot.thirdAngle;
                }
            }
            else {
                telemetry.addData("VuMark", "not visible");
                telemetry.update();
            }

            if (isVuforiaScanned == false)
            {
                Thread.sleep(2000);
                if (vuMark == RelicRecoveryVuMark.LEFT)
                {

                    telemetry.addData("VuMark", "Left");
                    telemetry.update();

                    int red = sensorColor.red();

                    int blue = sensorColor.blue();

                    if (blue<red)
                    {

                        encoder.backwardsEncoder (0.3, 300);

                        encoder.reset();

                        controlClass.liftArm();

                        encoder.forwardsEncoder (0.5, 4000);
```

```
                    encoder.reset();
                    encoder.turnRight (0.5, 450); //1440 right
turn
                    encoder.reset();
                    encoder.forwardsEncoder(0.5, 1000);
                    encoder.reset();
                    controlClass.openGrippers();
                    encoder.backwardsEncoder(0.5, 500);
                    isVuforiaScanned = true;
                } else
                {
                    encoder.forwardsEncoder (0.3, 400);
                    encoder.reset();
                    controlClass.liftArm();
                    encoder.forwardsEncoder (0.5, 3400);
                    encoder.reset();
                    encoder.turnRight (0.5, 400); //1440 right
turn
                    encoder.reset();
                    encoder.forwardsEncoder(0.5, 1000);
                    encoder.reset();
                    controlClass.openGrippers();
                    encoder.backwardsEncoder(0.5, 500);
                    isVuforiaScanned = true;
                }
            } else if (vuMark == RelicRecoveryVuMark.RIGHT)
            {
                telemetry.addData("VuMark", "Right");
                telemetry.update();
                int red = sensorColor.red();
```

```
                    int blue = sensorColor.blue();

                    if (blue<red)
                    {

                        encoder.backwardsEncoder (0.3, 300);

                        encoder.reset();

                        controlClass.liftArm();

                        encoder.forwardsEncoder (0.5, 4000);

                        encoder.reset();

                        encoder.turnRight (0.5, 1440); //1440 right
turn

                        encoder.reset();

                        encoder.forwardsEncoder(0.5, 2700);

                        encoder.reset();

                        encoder.turnLeft (0.5, 1440);

                        encoder.reset();

                        encoder.forwardsEncoder (0.5, 1000);

                        encoder.reset();

                        controlClass.openGrippers();

                        encoder.backwardsEncoder(0.5, 500);

                        isVuforiaScanned = true;

                    } else
                    {
                        encoder.forwardsEncoder (0.3, 400);

                        encoder.reset();

                        controlClass.liftArm();

                        encoder.forwardsEncoder (0.5, 3400);

                        encoder.reset();

                        encoder.turnRight (0.5, 1420); //1440 right
turn

                        encoder.reset();

                        encoder.forwardsEncoder(0.5, 2730);
```

```
                        encoder.reset();

                        encoder.turnLeft (0.5, 1440);

                        encoder.reset();

                        encoder.forwardsEncoder (0.5, 1000);

                        encoder.reset();

                        controlClass.openGrippers();

                        encoder.backwardsEncoder(0.5, 500);

                        isVuforiaScanned = true;


                    }


                } else if (vuMark == RelicRecoveryVuMark.CENTER ||
vuMark == RelicRecoveryVuMark.UNKNOWN)
                {

                    telemetry.addData("VuMark", "Center/Not Visible");
                    telemetry.update();

                    int red = sensorColor.red ();

                    int blue = sensorColor.blue ();

                    if (blue<red)
                    {

                        encoder.backwardsEncoder (0.3, 300);

                        encoder.reset();

                        controlClass.liftArm();

                        encoder.forwardsEncoder (0.5, 4000);

                        encoder.reset();

                        encoder.turnRight (0.5, 1440); //1440 right
turn

                        encoder.reset();

                        encoder.forwardsEncoder(0.5, 2000);

                        encoder.reset();

                        encoder.turnLeft (0.5, 1440);

                        encoder.reset();
```

```
                        encoder.forwardsEncoder (0.5, 1000);

                        encoder.reset();

                        controlClass.openGrippers();

                        encoder.backwardsEncoder(0.5, 500);

                        isVuforiaScanned = true;

                    } else
                    {

                        encoder.forwardsEncoder (0.3, 400);

                        encoder.reset();

                        controlClass.liftArm();

                        encoder.forwardsEncoder (0.5, 3400);

                        encoder.reset();

                        encoder.turnRight (0.5, 1440); //1440 right
turn

                        encoder.reset();

                        encoder.forwardsEncoder(0.5, 2000);

                        encoder.reset();

                        encoder.turnLeft (0.5, 1440);

                        encoder.reset();

                        encoder.forwardsEncoder (0.5, 1000);

                        encoder.reset();

                        controlClass.openGrippers();

                        encoder.backwardsEncoder(0.5, 500);

                        isVuforiaScanned = true;


                    }

                }

            }
         telemetry.update();
        }

    }
```

```
}
```