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

import com.qualcomm.robotcore.eventloop.opmode.Autonomous;
import com.qualcomm.robotcore.hardware.ColorSensor;
import org.firstinspires.ftc.robotcore.external.matrices.OpenGLMatrix;
import
org.firstinspires.ftc.robotcore.external.navigation.VuforiaLocalizer;
import org.firstinspires.ftc.robotcore.external.ClassFactory;
import
org.firstinspires.ftc.robotcore.external.navigation.VuforiaTrackables;
import
org.firstinspires.ftc.robotcore.external.navigation.VuforiaTrackable;
import
org.firstinspires.ftc.robotcore.external.navigation.RelicRecoveryVuMar
k;
import java.util.Locale;
import
org.firstinspires.ftc.robotcore.external.navigation.DistanceUnit;
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 com.qualcomm.robotcore.eventloop.opmode.LinearOpMode;
import com.qualcomm.robotcore.hardware.Gyroscope;
import com.qualcomm.robotcore.hardware.CRServo;
import com.qualcomm.robotcore.hardware.DcMotor;
import com.qualcomm.robotcore.hardware.DistanceSensor;
import com.qualcomm.robotcore.hardware.Servo;
import android.app.Activity;
import android.graphics.Color;
import android.view.View;

@Autonomous (name="Red Front", group="Red")

public class RedFront extends LinearOpMode{

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


    private DcMotor rightWheel = null;
    //asigns the veriable rightWheel dc motor propeties and sets it to
null
    private DcMotor leftWheel = null;
    //asigns the veriable leftWheel dc motor propeties and sets it to
null
    private DcMotor pully = null;
    //asigns the veriable pulley dc motor propeties and sets it to
null

    private Servo colorArm=null;
```

```
    //asigns the veriable colorArm servo propeties and sets it to null
    private CRServo glyphArm=null;
    //asigns the veriable glyphArm contious rotation servo propeties
and sets it to null


    ColorSensor sensorColor;
    //asigns the veriable sensorColor ColorSensor propeties and sets
it to null
    DistanceSensor sensorDistance;
    //asigns the veriable rigthWheel dc motor propeties and sets it to
null

    boolean colorRun=true;
    // sets the veriable colorRun to have only a true or false state

    OpenGLMatrix lastLocation = null;


    VuforiaLocalizer vuforia;

    @Override
    public void runOpMode() throws InterruptedException
    {

        //when you press init

        telemetry.addData("status", "Initialized");
        telemetry.update();


        rightWheel = hardwareMap.get(DcMotor.class, "rightWheel");
        //tells the variable to be assighned to the right motor
        leftWheel = hardwareMap.get(DcMotor.class, "leftWheel");
        //tells the variable to be assigned to the left motor
        leftWheel.setDirection(DcMotor.Direction.REVERSE);
        //sets the direction of the motor to reverse

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

        rightWheel.setDirection(DcMotor.Direction.FORWARD);
        //sets the direction of the motor to forward

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


        pully = hardwareMap.get(DcMotor.class, "pulley");
        //tells the variable to be assigned to the pulley motor
        pully.setZeroPowerBehavior(DcMotor.ZeroPowerBehavior.BRAKE);


        colorArm = hardwareMap.get(Servo.class, "colorArm");
        //tells the variable to be assigned to the color arm servo
        colorArm.setPosition(.3);
        //sets the position of the arm to be up right

        glyphArm = hardwareMap.get(CRServo.class, "glyphArm");
```

```
        //tells the variable to be assigned to the CR servo

        sensorColor=hardwareMap.get(ColorSensor.class, "colorSensor");
        //tells the variable to be assigned to the color/distance rev
sensor

        sensorDistance=hardwareMap.get(DistanceSensor.class,
"colorSensor");
        //tells the variable to be assigned to the color/distance rev
sensor







        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"); // can help in
debugging; otherwise not necessary

        telemetry.addData(">", "Press Play to start");
        telemetry.update();
        waitForStart();

        relicTrackables.activate();


        waitForStart();

        glyphArm.setPower(.3);
        //tells the cr servo to move inward
        Thread.sleep(2000);
        //wait for 2000 miliseconds
        glyphArm.setPower(.5);
        //stops the cr servo
        pully.setPower(-.5);
        //tells the pulley to move up at a .5 speed
        Thread.sleep(1000);
        //wait for 1000 miliseconds
        pully.setPower(0);
        //sets the power of the pully to upward at a .2 speed

        while (opModeIsActive()) {


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


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

                /* Found an instance of the template. In the actual
game, you will probably
                 * loop until this condition occurs, then move on to
act accordingly depending
                 * on which VuMark was visible. */
                telemetry.addData("VuMark", "%s visible", vuMark);

            } else {
                   telemetry.addData("VuMark", "not visible");
                    }

                    if (vuMark == RelicRecoveryVuMark.LEFT) {
                        telemetry.addData("VuMark", "LEFT
Successfull");
                        if (colorRun) {
```

```
                            colorArm.setPosition(.9);
                            //sets the servo at the jewl hight
                            Thread.sleep(500);
                            //wait for 500 miliseconds
                            int red=sensorColor.red();

                            int blue=sensorColor.green();

                            //addd data to display color sensor
information
                            telemetry.addData("Distance (cm)",
                                String.format(Locale.US, "%.02f",
sensorDistance.getDistance(DistanceUnit.CM)));
                            telemetry.addData("Alpha",
sensorColor.alpha());
                            telemetry.addData("Red  ",
sensorColor.red());
                            telemetry.addData("Green",
sensorColor.green());
                            telemetry.addData("Blue ",
sensorColor.blue());
                            telemetry.addData("Hue", hsvValues[0]);


                            telemetry.update();


                            Thread.sleep(100);
                            //wait for 100 miliseconds

                            if (red>blue) {
                                //if the red color is less than two

                                rightWheel.setPower(0);
                                //sets the power of the motor to 0
                                leftWheel.setPower(.3);
                                //sets the power of the motor to .3

                                //needs to be 90 degrees
                                Thread.sleep(550);
                                //wait for 600 miliseconds

                                colorArm.setPosition(.3);
                                //sets the position of the color arm
upwards


                                rightWheel.setPower(0);
                                //sets the power of the motor to 0
                                leftWheel.setPower(-.3);
                                //sets the power of the motor to -.3

                                Thread.sleep(640);
                                //wait for 640 miliseconds
                                rightWheel.setPower(.3);
                                //sets the power of the motor to .3
                                leftWheel.setPower(.3);
                                //sets the power of the motor to .3
```

```
Thread.sleep(2000);
//wait for 2500 miliseconds
rightWheel.setPower(0);
//sets the power of the motor to 0
leftWheel.setPower(0);
//sets the power of the motor to 0
rightWheel.setPower(.3);
//sets the power of the motor to -.3
leftWheel.setPower(-.3);
//sets the power of the motor to .3
Thread.sleep(2600);
//wait for 2500 miliseconds
rightWheel.setPower(.3);
//sets the power of the motor to .3
leftWheel.setPower(.3);
//sets the power of the motor to .3
Thread.sleep(2000);
//wait for 1500 miliseconds
rightWheel.setPower(0);
//sets the power of the motor to 0
leftWheel.setPower(0);
//sets the power of the motor to 0

glyphArm.setPower(.7);
//tells the glyph arm to move outwards

Thread.sleep(1000);
//wait for 1500 miliseconds
glyphArm.setPower(.3);
//tells the glyph arm to stop
Thread.sleep(1000);

glyphArm.setPower(.5);

rightWheel.setPower(-.3);
leftWheel.setPower(-.3);

Thread.sleep(300);

rightWheel.setPower(0);
leftWheel.setPower(0);

pully.setPower(.3);

Thread.sleep(1000);

pully.setPower(0);

rightWheel.setPower(.3);
leftWheel.setPower(.3);

Thread.sleep(500);

rightWheel.setPower(0);
leftWheel.setPower(0);
```

```
                } else {

                    rightWheel.setPower(-.3);
                    //sets the power of the motor to 0
                    leftWheel.setPower(0);
                    //sets the power of the motor to .3

                    //needs to be 90 degrees
                    Thread.sleep(550);
                    //wait for 600 miliseconds

                    colorArm.setPosition(.3);
                    //sets the position of the color arm
upwards

                    rightWheel.setPower(.3);
                    //sets the power of the motor to 0
                    leftWheel.setPower(0);
                    //sets the power of the motor to -.3

                    Thread.sleep(640);
                    //wait for 640 miliseconds
                    rightWheel.setPower(.3);
                    //sets the power of the motor to .3
                    leftWheel.setPower(.3);
                    //sets the power of the motor to .3
                    Thread.sleep(2000);
                    //wait for 2500 miliseconds
                    rightWheel.setPower(0);
                    //sets the power of the motor to 0
                    leftWheel.setPower(0);
                    //sets the power of the motor to 0
                    rightWheel.setPower(.3);
                    //sets the power of the motor to -.3
                    leftWheel.setPower(-.3);
                    //sets the power of the motor to .3
                    Thread.sleep(2600);
                    //wait for 2500 miliseconds
                    rightWheel.setPower(.3);
                    //sets the power of the motor to .3
                    leftWheel.setPower(.3);
                    //sets the power of the motor to .3
                    Thread.sleep(2000);
                    //wait for 1500 miliseconds
                    rightWheel.setPower(0);
                    //sets the power of the motor to 0
                    leftWheel.setPower(0);
                    //sets the power of the motor to 0

                    glyphArm.setPower(.7);
                    //tells the glyph arm to move outwards

                    Thread.sleep(1000);
                    //wait for 1500 miliseconds
                    glyphArm.setPower(.3);
                    //tells the glyph arm to stop
```

```
                                Thread.sleep(1000);

                                glyphArm.setPower(.5);

                                rightWheel.setPower(-.3);
                                leftWheel.setPower(-.3);

                                Thread.sleep(300);

                                rightWheel.setPower(0);
                                leftWheel.setPower(0);

                                pully.setPower(.3);

                                Thread.sleep(1000);

                                pully.setPower(0);

                                rightWheel.setPower(.3);
                                leftWheel.setPower(.3);

                                Thread.sleep(500);

                                rightWheel.setPower(0);
                                leftWheel.setPower(0);

                            }


                            leftWheel.setPower(0);
                            rightWheel.setPower(0);


                            colorRun=false;

                        }
                    }


            //rightWheel.setPower(.5); leftWheel.setPower(.5);

            //Thread.sleep(100);


            //rightWheel.setPower(0); leftWheel.setPower(0);


                     else if (vuMark == RelicRecoveryVuMark.RIGHT) {
                        telemetry.addData("VuMark", "RIGHT
Sucessful");
                            if (colorRun) {


                            colorArm.setPosition(.9);

                            Thread.sleep(1000);
```

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

                            int blue=sensorColor.green();


                            telemetry.addData("Distance (cm)",
                                String.format(Locale.US, "%.02f",
sensorDistance.getDistance(DistanceUnit.CM)));
                            telemetry.addData("Alpha",
sensorColor.alpha());
                            telemetry.addData("Red  ",
sensorColor.red());
                            telemetry.addData("Green",
sensorColor.green());
                            telemetry.addData("Blue ",
sensorColor.blue());
                            telemetry.addData("Hue", hsvValues[0]);


                            telemetry.update();


                            Thread.sleep(100);


                            if (red>blue) {


                                rightWheel.setPower(0);
                                //sets the power of the motor to 0
                                leftWheel.setPower(.3);
                                //sets the power of the motor to .3

                                //needs to be 90 degrees
                                Thread.sleep(550);
                                //wait for 600 miliseconds

                                colorArm.setPosition(.3);
                                //sets the position of the color arm
upwards

                                rightWheel.setPower(0);
                                //sets the power of the motor to 0
                                leftWheel.setPower(-.3);
                                //sets the power of the motor to -.3

                                Thread.sleep(640);
                                //wait for 640 miliseconds
                                rightWheel.setPower(.3);
                                //sets the power of the motor to .3
                                leftWheel.setPower(.3);
                                //sets the power of the motor to .3
                                Thread.sleep(2000);
                                //wait for 2500 miliseconds
                                rightWheel.setPower(0);
                                //sets the power of the motor to 0
                                leftWheel.setPower(0);
```

```
                    //sets the power of the motor to 0
                    rightWheel.setPower(.3);
                    //sets the power of the motor to -.3
                    leftWheel.setPower(-.3);
                    //sets the power of the motor to .3
                    Thread.sleep(2000);
                    //wait for 2500 miliseconds
                    rightWheel.setPower(.3);
                    //sets the power of the motor to .3
                    leftWheel.setPower(.3);
                    //sets the power of the motor to .3
                    Thread.sleep(1500);
                    //wait for 1500 miliseconds
                    rightWheel.setPower(0);
                    //sets the power of the motor to 0
                    leftWheel.setPower(0);
                    //sets the power of the motor to 0

                    glyphArm.setPower(.7);
                    //tells the glyph arm to move outwards

                    Thread.sleep(1000);
                    //wait for 1500 miliseconds
                    glyphArm.setPower(.3);
                    //tells the glyph arm to stop
                    Thread.sleep(1000);

                    glyphArm.setPower(.5);

                    rightWheel.setPower(-.3);
                    leftWheel.setPower(-.3);

                    Thread.sleep(300);

                    rightWheel.setPower(0);
                    leftWheel.setPower(0);

                    pully.setPower(.3);

                    Thread.sleep(1000);

                    rightWheel.setPower(.3);
                    leftWheel.setPower(.3);

                    Thread.sleep(500);

                    rightWheel.setPower(0);
                    leftWheel.setPower(0);

                } else {


                     rightWheel.setPower(0);
                    //sets the power of the motor to 0
                    leftWheel.setPower(-.3);
                    //sets the power of the motor to .3
```

```
                                    //needs to be 90 degrees
                                    Thread.sleep(550);
                                    //wait for 600 miliseconds

                                    colorArm.setPosition(.3);
                                    //sets the position of the color arm
upwards

                                    rightWheel.setPower(0);
                                    //sets the power of the motor to 0
                                    leftWheel.setPower(.3);
                                    //sets the power of the motor to -.3

                                    Thread.sleep(640);
                                    //wait for 640 miliseconds
                                    rightWheel.setPower(.3);
                                    //sets the power of the motor to .3
                                    leftWheel.setPower(.3);
                                    //sets the power of the motor to .3
                                    Thread.sleep(2000);
                                    //wait for 2500 miliseconds
                                    rightWheel.setPower(0);
                                    //sets the power of the motor to 0
                                    leftWheel.setPower(0);
                                    //sets the power of the motor to 0
                                    rightWheel.setPower(.3);
                                    //sets the power of the motor to -.3
                                    leftWheel.setPower(-.3);
                                    //sets the power of the motor to .3
                                    Thread.sleep(2500);
                                    //wait for 2500 miliseconds
                                    rightWheel.setPower(.3);
                                    //sets the power of the motor to .3
                                    leftWheel.setPower(.3);
                                    //sets the power of the motor to .3
                                    Thread.sleep(2500);
                                    //wait for 1500 miliseconds
                                    rightWheel.setPower(0);
                                    //sets the power of the motor to 0
                                    leftWheel.setPower(0);
                                    //sets the power of the motor to 0

                                    glyphArm.setPower(.7);
                                    //tells the glyph arm to move outwards

                                    Thread.sleep(1000);
                                    //wait for 1500 miliseconds
                                    glyphArm.setPower(.3);
                                    //tells the glyph arm to stop
                                    Thread.sleep(1000);

                                    glyphArm.setPower(.5);

                                    rightWheel.setPower(-.3);
                                    leftWheel.setPower(-.3);

                                    Thread.sleep(300);
```

```
                                rightWheel.setPower(0);
                                leftWheel.setPower(0);

                                pully.setPower(.3);

                                Thread.sleep(1000);

                                pully.setPower(0);

                                rightWheel.setPower(.3);
                                leftWheel.setPower(.3);

                                Thread.sleep(500);

                                rightWheel.setPower(0);
                                leftWheel.setPower(0);

                            }


                            leftWheel.setPower(0);
                            rightWheel.setPower(0);


                            colorArm.setPosition(.3);


                            colorRun=false;

                        }




            //rightWheel.setPower(.5); leftWheel.setPower(.5);

            //Thread.sleep(100);


            //rightWheel.setPower(0); leftWheel.setPower(0);


                    } else if (vuMark == RelicRecoveryVuMark.CENTER ||
                     vuMark == RelicRecoveryVuMark.UNKNOWN) {
                        if (vuMark == RelicRecoveryVuMark.CENTER)
                            telemetry.addData("VuMark", "CENTER
Sucessful");
                        if (vuMark == RelicRecoveryVuMark.UNKNOWN)
                            telemetry.addData("VuMark", "UNKNOWN");
                        if (colorRun) {


                            colorArm.setPosition(.9);

                            Thread.sleep(1000);
```

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

                            int blue=sensorColor.green();


                            telemetry.addData("Distance (cm)",
                                String.format(Locale.US, "%.02f",
sensorDistance.getDistance(DistanceUnit.CM)));
                            telemetry.addData("Alpha",
sensorColor.alpha());
                            telemetry.addData("Red  ",
sensorColor.red());
                            telemetry.addData("Green",
sensorColor.green());
                            telemetry.addData("Blue ",
sensorColor.blue());
                            telemetry.addData("Hue", hsvValues[0]);


                            telemetry.update();


                            Thread.sleep(100);


                            if (red>blue) {


                                rightWheel.setPower(0);
                                //sets the power of the motor to 0
                                leftWheel.setPower(.3);
                                //sets the power of the motor to .3

                                //needs to be 90 degrees
                                Thread.sleep(550);
                                //wait for 600 miliseconds

                                colorArm.setPosition(.3);
                                //sets the position of the color arm
upwards


                                rightWheel.setPower(0);
                                //sets the power of the motor to 0
                                leftWheel.setPower(-.3);
                                //sets the power of the motor to -.3

                                Thread.sleep(640);
                                //wait for 640 miliseconds
                                rightWheel.setPower(.3);
                                //sets the power of the motor to .3
                                leftWheel.setPower(.3);
                                //sets the power of the motor to .3
                                Thread.sleep(2000);
                                //wait for 2500 miliseconds
                                rightWheel.setPower(0);
                                //sets the power of the motor to 0
```

```
                leftWheel.setPower(0);
                //sets the power of the motor to 0
                rightWheel.setPower(.3);
                //sets the power of the motor to -.3
                leftWheel.setPower(-.3);
                //sets the power of the motor to .3
                Thread.sleep(2000);
                //wait for 2500 miliseconds
                rightWheel.setPower(.3);
                //sets the power of the motor to .3
                leftWheel.setPower(.3);
                //sets the power of the motor to .3
                Thread.sleep(1500);
                //wait for 1500 miliseconds
                rightWheel.setPower(0);
                //sets the power of the motor to 0
                leftWheel.setPower(0);
                //sets the power of the motor to 0

                glyphArm.setPower(.7);
                //tells the glyph arm to move outwards

                Thread.sleep(1000);
                //wait for 1500 miliseconds
                glyphArm.setPower(.3);
                //tells the glyph arm to stop
                Thread.sleep(1000);

                glyphArm.setPower(.5);

                rightWheel.setPower(-.3);
                leftWheel.setPower(-.3);

                Thread.sleep(300);

                rightWheel.setPower(0);
                leftWheel.setPower(0);

                pully.setPower(.3);

                Thread.sleep(1000);

                rightWheel.setPower(.3);
                leftWheel.setPower(.3);

                Thread.sleep(500);

                rightWheel.setPower(0);
                leftWheel.setPower(0);

            } else {


                 rightWheel.setPower(0);
                //sets the power of the motor to 0
                leftWheel.setPower(-.3);
                //sets the power of the motor to .3
```

```
                                    //needs to be 90 degrees
                                    Thread.sleep(550);
                                    //wait for 600 miliseconds

                                    colorArm.setPosition(.3);
                                    //sets the position of the color arm
upwards

                                    rightWheel.setPower(0);
                                    //sets the power of the motor to 0
                                    leftWheel.setPower(.3);
                                    //sets the power of the motor to -.3

                                    Thread.sleep(640);
                                    //wait for 640 miliseconds
                                    rightWheel.setPower(.3);
                                    //sets the power of the motor to .3
                                    leftWheel.setPower(.3);
                                    //sets the power of the motor to .3
                                    Thread.sleep(2000);
                                    //wait for 2500 miliseconds
                                    rightWheel.setPower(0);
                                    //sets the power of the motor to 0
                                    leftWheel.setPower(0);
                                    //sets the power of the motor to 0
                                    rightWheel.setPower(.3);
                                    //sets the power of the motor to -.3
                                    leftWheel.setPower(-.3);
                                    //sets the power of the motor to .3
                                    Thread.sleep(2500);
                                    //wait for 2500 miliseconds
                                    rightWheel.setPower(.3);
                                    //sets the power of the motor to .3
                                    leftWheel.setPower(.3);
                                    //sets the power of the motor to .3
                                    Thread.sleep(2500);
                                    //wait for 1500 miliseconds
                                    rightWheel.setPower(0);
                                    //sets the power of the motor to 0
                                    leftWheel.setPower(0);
                                    //sets the power of the motor to 0

                                    glyphArm.setPower(.7);
                                    //tells the glyph arm to move outwards

                                    Thread.sleep(1000);
                                    //wait for 1500 miliseconds
                                    glyphArm.setPower(.3);
                                    //tells the glyph arm to stop
                                    Thread.sleep(1000);

                                    glyphArm.setPower(.5);

                                    rightWheel.setPower(-.3);
                                    leftWheel.setPower(-.3);
```

```
                                Thread.sleep(300);

                                rightWheel.setPower(0);
                                leftWheel.setPower(0);

                                pully.setPower(.3);

                                Thread.sleep(1000);

                                pully.setPower(0);

                                rightWheel.setPower(.3);
                                leftWheel.setPower(.3);

                                Thread.sleep(500);

                                rightWheel.setPower(0);
                                leftWheel.setPower(0);

                            }


                            leftWheel.setPower(0);
                            rightWheel.setPower(0);


                            colorArm.setPosition(.3);


                            colorRun=false;

                        }



            //rightWheel.setPower(.5); leftWheel.setPower(.5);

            //Thread.sleep(100);


            //rightWheel.setPower(0); leftWheel.setPower(0);
                    } else
                        telemetry.addData("VuMark", "YOU FAIL!!!!");

                /* For fun, we also exhibit the navigational pose. In
the Relic Recovery game,
                 * it is perhaps unlikely that you will actually need
to act on this pose information, but
                 * we illustrate it nevertheless, for completeness. */
                OpenGLMatrix pose =
((VuforiaTrackableDefaultListener)relicTemplate.getListener()).getPose
();
                telemetry.addData("Pose", format(pose));

                /* We further illustrate how to decompose the pose
into useful rotational and
```

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

                    // Extract the X, Y, and Z components of the
offset of the target relative to the robot
                    double tX = trans.get(0);
                    double tY = trans.get(1);
                    double tZ = trans.get(2);

                    // Extract the rotational components of the target
relative to the robot
                    double rX = rot.firstAngle;
                    double rY = rot.secondAngle;
                    double rZ = rot.thirdAngle;
                }




            telemetry.update();
        }
    }

    String format(OpenGLMatrix transformationMatrix) {
        return (transformationMatrix != null) ?
transformationMatrix.formatAsTransform() : "null";
    }

}
```