Red Double Skystone Program

```
1  package org.firstinspires.ftc.teamcode;
2
3  import com.qualcomm.hardware.bosch.BNO055IMU;
4  import com.qualcomm.robotcore.eventloop.opmode.Autonomous;
5  import com.qualcomm.robotcore.eventloop.opmode.Disabled;
6  import com.qualcomm.robotcore.eventloop.opmode.LinearOpMode;
7  import com.qualcomm.robotcore.hardware.CRServo;
8  import com.qualcomm.robotcore.hardware.DcMotor;
9  import com.qualcomm.robotcore.hardware.DcMotorSimple;
10 import com.qualcomm.robotcore.hardware.Servo;
11 import com.qualcomm.robotcore.util.ElapsedTime;
12
13 import org.firstinspires.ftc.robotcore.external.ClassFactory;
14 import org.firstinspires.ftc.robotcore.external.hardware.
   camera.WebcamName;
15 import org.firstinspires.ftc.robotcore.external.navigation.
   VuforiaLocalizer;
16 import org.firstinspires.ftc.robotcore.external.tfod.
   Recognition;
17 import org.firstinspires.ftc.robotcore.external.tfod.
   TFObjectDetector;
18
19 import java.util.List;
20
21 @Autonomous (name="Red Double", group="Red")
22
23 public class RedDouble extends LinearOpMode
24 {
25
26     private static final String TFOD_MODEL_ASSET = "Skystone.
   tflite";
27     private static final String LABEL_FIRST_ELEMENT = "Stone";
28     private static final String LABEL_SECOND_ELEMENT = "
   Skystone";
29
30     /*
31      * IMPORTANT: You need to obtain your own license key to
   use Vuforia. The string below with which
32      * 'parameters.vuforiaLicenseKey' is initialized is for
   illustration only, and will not function.
33      * A Vuforia 'Development' license key, can be obtained
   free of charge from the Vuforia developer
```

```
34      * web site at https://developer.vuforia.com/license-
   manager.
35      *
36      * Vuforia license keys are always 380 characters long,
   and look as if they contain mostly
37      * random data. As an example, here is a example of a
   fragment of a valid key:
38      *      ...
   yIgIzTqZ4mWjk9wd3cZO9T1axEqzuhxoGlfOOI2dRzKS4T0hQ8kT ...
39      * Once you've obtained a license key, copy the string
   from the Vuforia web site
40      * and paste it in to your code on the next line, between
   the double quotes.
41      */
42     private static final String VUFORIA_KEY = "AQUWr4X/////
   AAABme38EPssRkvls9+q/
   BGPYgxKXBXELWHMdkTcCqUqHeyDpyXGWFLCTABgDXEMGe1EmsnDQxmJ7WQ069J
   3YSv+kOcfq3g2EnwZr2O3DujsIU1nT0aXgLlAtQU2r7wWAgHvR9ADO5pe/
   q7MzCyhjSTQLCgizGFLgmqfre0A9rjYcXYbYw11R3P7VRHnL3QHn3QH2oFVQfM
   b+dIzmZkfv0cd5qWvdhjovYF8hpZ/
   HT7veIa8ZQ9CIQ0541pxplXVud80z1xWpjFGJPaoQGO+xKWZ8E+
   Zlu7z5umiaV1+
   ChGeJ9pPyIJn0LsnoIHumZoYb4di4tFygMPVmH8ChsTlGJjaPBSCRBFjxzBqsX
   mBZY7eCa6S";
43
44     /**
45      * {@link #vuforia} is the variable we will use to store
   our instance of the Vuforia
46      * localization engine.
47      */
48     private VuforiaLocalizer vuforia;
49
50     /**
51      * {@link #tfod} is the variable we will use to store our
   instance of the TensorFlow Object
52      * Detection engine.
53      */
54     private TFObjectDetector tfod;
55
56     //Wheel Motors
57     DcMotor leftFront;
58     DcMotor rightFront;
```

```
59      DcMotor leftBack;
60      DcMotor rightBack;
61
62      //Foundation Servos
63      Servo leftFoundation;
64      Servo rightFoundation;
65
66      //Turret Motor
67      DcMotor turret;
68
69      //Lift Motor
70      DcMotor lift;
71
72      //Arm Motor
73      DcMotor arm;
74
75      //Claw Continuous Rotation Servo
76      CRServo leftClaw;
77      CRServo rightClaw;
78
79      Servo capstone;
80
81      //Gyroscope
82      BNO055IMU imu;
83
84      //Robot Classes
85      DriveTrain dT;
86      Armstrong a;
87
88      //State Machine Class
89      RedStates rS;
90
91      //Timer
92      ElapsedTime runtime;
93
94      //Stone counter
95      int count = 1;
96      //Extra distance for each stone
97      int extraTick;
98
99      //Boolean for if the skystone has been found
100     boolean skystone = false;
```

```
101
102     //The label of the object the program sees
103     String label;
104
105     @Override
106     public void runOpMode() throws InterruptedException
107     {
108
109         // The TFObjectDetector uses the camera frames from
    the VuforiaLocalizer, so we create that
110         // first.
111         initVuforia();
112
113         if (ClassFactory.getInstance().
    canCreateTFObjectDetector()) {
114             initTfod();
115         } else {
116             telemetry.addData("Sorry!", "This device is not
    compatible with TFOD");
117         }
118
119         /**
120          * Activate TensorFlow Object Detection before we
    wait for the start command.
121          * Do it here so that the Camera Stream window will
    have the TensorFlow annotations visible.
122          **/
123         if (tfod != null) {
124             tfod.activate();
125         }
126
127         //Setting variables to the real life components using
     the configuration on the phone.
128         leftFront = hardwareMap.dcMotor.get("leftFront");
129         rightFront = hardwareMap.dcMotor.get("rightFront");
130         leftBack = hardwareMap.dcMotor.get("leftBack");
131         rightBack = hardwareMap.dcMotor.get("rightBack");
132
133         leftFoundation = hardwareMap.servo.get("
    leftFoundation");
134         rightFoundation = hardwareMap.servo.get("
    rightFoundation");
```

```
135
136         turret = hardwareMap.dcMotor.get("turret");
137
138         lift = hardwareMap.dcMotor.get("lift");
139
140         arm = hardwareMap.dcMotor.get("arm");
141
142         leftClaw = hardwareMap.crservo.get("leftClaw");
143         rightClaw = hardwareMap.crservo.get("rightClaw");
144
145         capstone = hardwareMap.servo.get("capstone");
146
147         imu = hardwareMap.get(BNO055IMU.class, "imu");
148
149         //Setting up the classes to run using the variables
    above
150         dT = new DriveTrain(gamepad1, gamepad2 ,leftFront,
    rightFront, leftBack, rightBack, leftFoundation,
    rightFoundation);
151         a = new Armstrong(gamepad1, gamepad2,  turret, lift,
    arm, leftClaw, rightClaw, capstone, imu);
152         runtime = new ElapsedTime();
153
154         //Set the motors to stop and stay still instead of
    removing all power and coasting
155         leftFront.setZeroPowerBehavior(DcMotor.
    ZeroPowerBehavior.BRAKE);
156         leftFront.setDirection(DcMotorSimple.Direction.
    REVERSE);
157
158         rightFront.setZeroPowerBehavior(DcMotor.
    ZeroPowerBehavior.BRAKE);
159
160         leftBack.setZeroPowerBehavior(DcMotor.
    ZeroPowerBehavior.BRAKE);
161         leftBack.setDirection(DcMotorSimple.Direction.REVERSE
    );
162
163         rightBack.setZeroPowerBehavior(DcMotor.
    ZeroPowerBehavior.BRAKE);
164
165         turret.setZeroPowerBehavior(DcMotor.ZeroPowerBehavior
```

Red Double Skystone Program

```
165 .BRAKE);
166
167         lift.setZeroPowerBehavior(DcMotor.ZeroPowerBehavior.
    BRAKE);
168
169         arm.setZeroPowerBehavior(DcMotor.ZeroPowerBehavior.
    BRAKE);
170
171         //Stops Robot
172         dT.kill();
173         dT.release();
174         a.kill();
175
176         //Setting up Red States
177         rS = new RedStates(leftFront, rightFront, leftBack,
    rightBack, leftFoundation, rightFoundation, turret, lift, arm
    , leftClaw, rightClaw, dT, a, runtime);
178
179         /** Wait for the game to begin */
180         telemetry.addData(">", "Press Play to start op mode")
    ;
181         telemetry.update();
182
183         waitForStart();
184
185         //Resets Timer
186         runtime.reset();
187
188         //If the play button is pressed
189         //Uncomment to activate skystone detection
190         if (opModeIsActive())
191         {
192
193             //While the state machine is still running
194             while (rS.getState() != "finished")
195             {
196
197                 //Run the state machine method
198                 rS.runFull();
199
200                 //Telemetry that displays the current state
201                 telemetry.addData("State>", rS.getState());
```

```
202
203                 //Display Telemetry
204                 telemetry.update();
205
206                 arm.setPower(0);
207
208             }
209
210             //Loops the program
211             while (opModeIsActive())
212             {
213
214                 if (tfod != null)
215                 {
216
217                     // getUpdatedRecognitions() will return
    null if no new information is available since
218                     // the last time that call was made.
219
220                     List<Recognition> updatedRecognitions =
    tfod.getUpdatedRecognitions();
221
222                     if (updatedRecognitions != null)
223                     {
224
225                         telemetry.addData("# Object Detected"
    , updatedRecognitions.size());
226
227                         // step through the list of
    recognitions and display boundary info.
228
229                         int i = 0;
230
231                         for (Recognition recognition :
    updatedRecognitions)
232                         {
233
234                             telemetry.addData(String.format("
    label (%d)", i), recognition.getLabel());
235                             //Sets the label to the name of
    the recognition
236                             label = recognition.getLabel();
```

```
237
238                             telemetry.addData(String.format(
    "  left,top (%d)", i), "%.03f , %.03f",
239                                     recognition.getLeft(),
    recognition.getTop());
240
241                             telemetry.addData(String.format(
    "  right,bottom (%d)", i), "%.03f , %.03f",
242                                     recognition.getRight(),
    recognition.getBottom());
243
244                             telemetry.update();
245                         }
246
247                         //If label equals Skystone or if two
    stones have been detected
248                         if(label == "Skystone" || count == 3)
249                         {
250
251                             //Telemetry for detecting the
    Skystone
252                             telemetry.addData("Guess What", "
    Skystone Baby");
253
254                             //Displays telemetry
255                             telemetry.update();
256
257                             //Sets the Skystone flag to true
258                             skystone = true;
259
260                             //Stops robot
261                             dT.kill();
262
263                             //Exits while loop
264                             break;
265
266                             //If the label detects Stone and
    the timer has been going for one second
267                         } else if (label == "Stone" &&
    runtime.milliseconds() >= 1000)
268                         {
269
```

```
270                             //Telemetry for not seeing a
    Skystone
271                             telemetry.addData("Guess What", "
    Nothing!");
272
273                             //Displays Telemetry
274                             telemetry.update();
275
276                             //Adds one to the count number
277                             count++;
278
279                             //Adds to the extra tick distance
280                             extraTick = extraTick + 380;
281
282                             //Moves left to next stone
283                             dT.left(0.25, 380);
284
285                             //Resets timer
286                             runtime.reset();
287
288                             //If the robot times out
289                         } else if (runtime.milliseconds() >=
    1500)
290                         {
291
292                             //Rotate arm up to create
    movement to help detection
293                             a.rUp(0.25, 50);
294
295                             //Resets timer
296                             runtime.reset();
297
298                         }
299
300                         //Displays Telemetry
301                         telemetry.update();
302
303                     }
304                 }
305             }
306         }
307
```

Red Double Skystone Program

```
308         if (skystone == true)
309         {
310
311             //Moves robot forward to the skystone
312             dT.forward(0.25, 350);
313
314             //Grabs Skystone
315             leftClaw.setPower(-1);
316             rightClaw.setPower(1);
317
318             //Resets Timer
319             runtime.reset();
320
321             //Waits a quarter second to clamp
322             while (runtime.milliseconds() < 250);
323
324             //Rotates arm up to a level position
325             a.rUp(0.5, 300);
326
327             //Sets idle power
328             arm.setPower(0.1);
329             leftClaw.setPower(-1);
330             rightClaw.setPower(1);
331
332             //Moves robot back to clear the bridge structure
333             dT.backwards(0.3, 350);
334
335             //Moves robot right to the foundation plate
336             dT.right(0.25, 2000 + extraTick);
337
338             //Moves the robot forward to move stone out of
    the way
339             dT.forward(0.4, 250);
340
341             a.setCIdle(0);
342
343             a.unclamp(1, 250);
344
345             arm.setPower(0);
346
347             //Moves robot clear of bridge
348             dT.backwards(0.4, 250);
```

```
349
350             dT.left(0.25,3140 + extraTick);
351
352             dT.forward(0.25, 300);
353
354             leftClaw.setPower(-1);
355             rightClaw.setPower(1);
356
357             a.rUp(0.25, 250);
358
359             arm.setPower(0.1);
360             leftClaw.setPower(-1);
361             rightClaw.setPower(1);
362
363             dT.backwards(0.25, 250);
364
365             dT.right(0.25, 3140 +extraTick);
366
367             a.unclamp(1, 250);
368
369             dT.left(0.4, 500);
370
371         }
372
373     }
374
375     /**
376      * Initialize the Vuforia localization engine.
377      */
378     private void initVuforia() {
379         /*
380          * Configure Vuforia by creating a Parameter object,
    and passing it to the Vuforia engine.
381          */
382         VuforiaLocalizer.Parameters parameters = new
    VuforiaLocalizer.Parameters();
383
384         parameters.vuforiaLicenseKey = VUFORIA_KEY;
385         parameters.cameraName = hardwareMap.get(WebcamName.
    class, "Webcam 1");
386
387         //  Instantiate the Vuforia engine
```

```
388         vuforia = ClassFactory.getInstance().createVuforia(
    parameters);
389
390         // Loading trackables is not necessary for the
    TensorFlow Object Detection engine.
391     }
392
393     /**
394      * Initialize the TensorFlow Object Detection engine.
395      */
396     private void initTfod() {
397         int tfodMonitorViewId = hardwareMap.appContext.
    getResources().getIdentifier(
398                 "tfodMonitorViewId", "id", hardwareMap.
    appContext.getPackageName());
399         TFObjectDetector.Parameters tfodParameters = new
    TFObjectDetector.Parameters(tfodMonitorViewId);
400         tfodParameters.minimumConfidence = 0.8;
401         tfod = ClassFactory.getInstance().
    createTFObjectDetector(tfodParameters, vuforia);
402         tfod.loadModelFromAsset(TFOD_MODEL_ASSET,
    LABEL_FIRST_ELEMENT, LABEL_SECOND_ELEMENT);
403     }
404
405 }
```