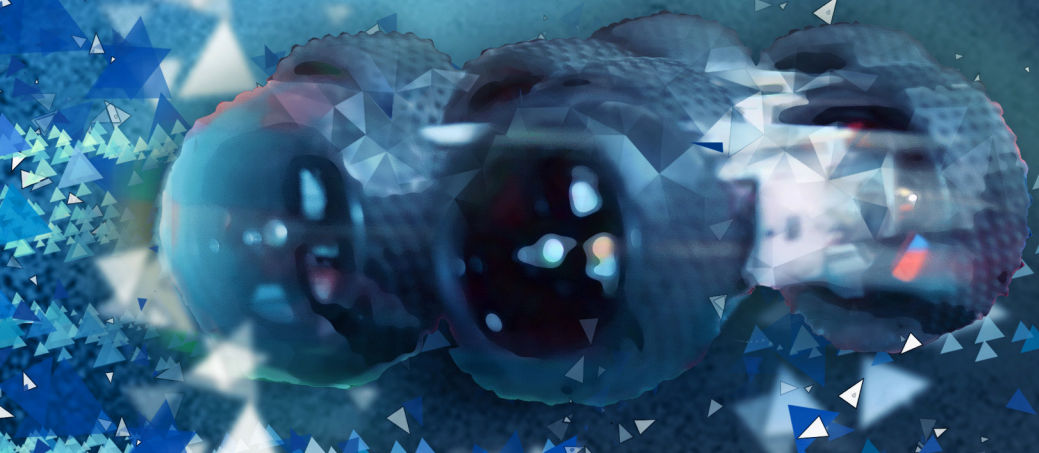
Rolls-Royce

GETTING STARTED GUIDE
WITH
RACE YOUR CODE

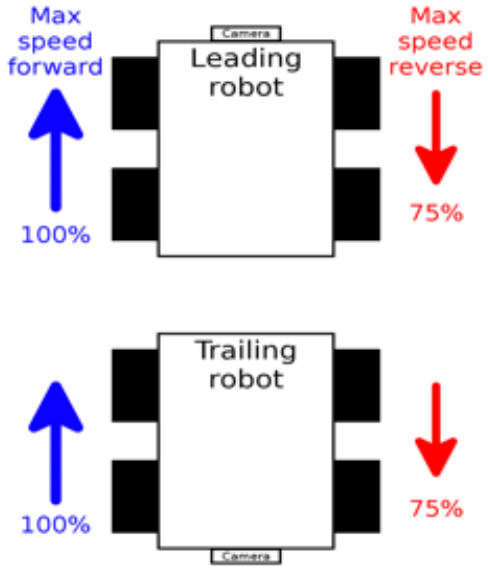


The below is insight from the people who create and build the robots you'll be racing on how the challenge works and the code you will receive.

THE CHALLENGE

RaceYourCode is all about taking a pair of pre-programmed robots which can navigate our race track on their own, and improving the code so that the robots get the best possible lap time you can achieve. This could be tricky as your only sensor input is from a camera mounted at the front of both robots. Improvements can be as simple as changing the lane on the track or as complicated as major changes such as a complete code rewrite.

So why are there two robots? Because one has to navigate the track backwards!



The speed limiting means that whilst you could run the trailing robot forwards, it will not be as competitive as it will move about 33% faster when running backwards as intended. The robots send telemetry data to each other as well so they know what the other one is doing at all times. As the front robot has a better view of the track, it is quicker and more consistent at navigating the lap. There are also subtle differences in the speeds that each robot does when commanded to 100%.

This leaves a few obvious places to start for making improvements:

1. Using data from the leading robot to improve the trailing robot's understanding of the track ahead
2. Plot a better line around the track, possibly different for each robot
3. Tweaking the settings so that the robots are better controlled
4. Replace the camera processing with something faster or more accurate
5. Compensating for differences in front and rear robot lap times

But before we can improve the code we need to understand what we already have.

STEPS FOR COMPETITORS

We recommend two monitors or machines, one for the Twitch stream and one for the two VNC sessions. It is possible to use a single monitor, but it will be more difficult to control and watch at the same time.

Test you can view the stream prior to your race time

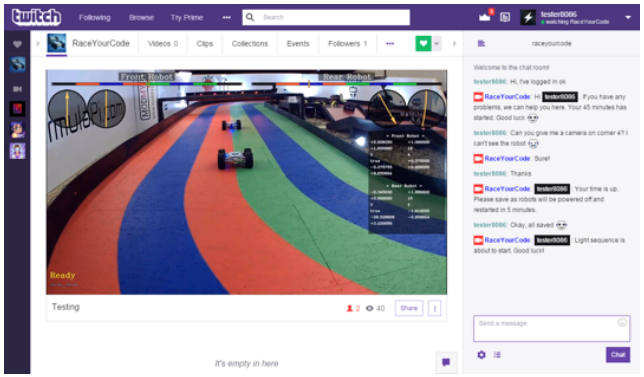
You should be able to test you are successfully viewing Twitch streams by viewing a major Twitch channel. For example https://www.twitch.tv/twitch

If you can successfully watch a Twitch stream, you should be ready to watch the raceyourcode stream when it becomes available. It is not necessary to set up a Twitch account, but it is recommended as you will need this in order to participate in the chat window.

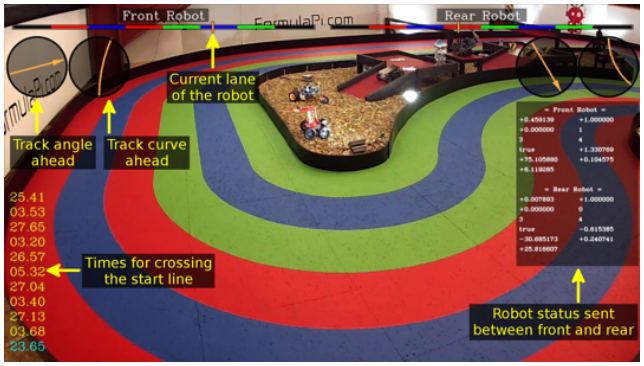
View the stream

The raceyourcode stream will be live 5-10 minutes before your race time. You can view the stream at https://www.twitch.tv/raceyourcode

The stream is delayed by about 10 seconds, so keep this in mind when issuing commands and waiting for results.



Explanation of the stream image telemetry



STEPS FOR COMPETITORS

Testing your VNC connection works prior to your race time

Controlling the robots is done via a VNC connection. To do this you will need a VNC client.

Most VNC clients should work, but we recommend using TightVNC which is available at http://www.tightvnc.com/

To install this on Windows:

http://www.tightvnc.com/download.php

Choose 32 or 64 bit to match your version of Windows

You do not need to install the server parts, just the client / viewer

Run the TightVNC Viewer from the start menu

Connect to 217.36.211.159:3

To install this on Ubuntu:

Install with:
sudo apt-get install xtightvncviewer

Open a viewer:
xtightvncviewer 217.36.211.159:3

To install on Mac:

As TightVNC is not available for Mac, try installing RealVNC

https://www.realvnc.com/download/viewer/macos/

Connect to 217.36.211.159:3

If you are successfully logged in you should see the following screen:

Password is ryctest1

If you are successfully logged in you should see the following screen:

STEPS FOR COMPETITORS

Logging in to the robots

To log in to the robots during your slot, you will need to open two separate VNC connections: One VNC connection to the front robot, and a separate VNC connection to the rear robot.

Windows:

Run the TightVNC Viewer from the start menu twice (one for each robot)

Connect to 217.36.211.159:1 and 217.36.211.159:2

To install this on Ubuntu:

Open two viewers (one for each robot):

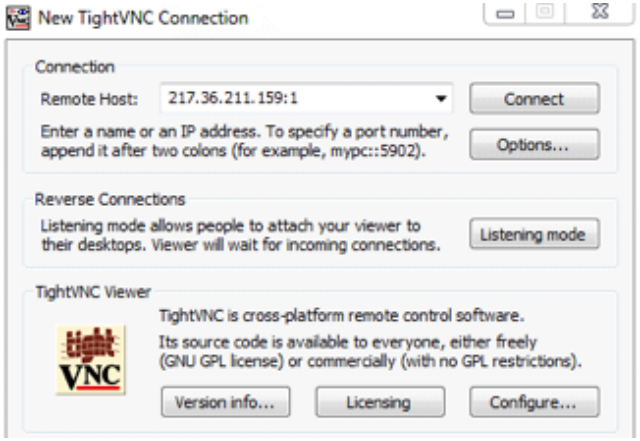
xtightvncviewer 217.36.211.159:1

xtightvncviewer 217.36.211.159:2

To install on Mac:

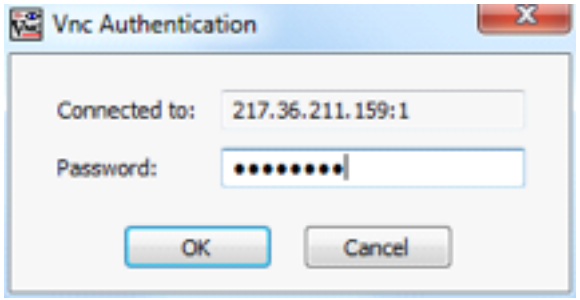
Run RealVNC viewer twice (one for each robot)

Connect to 217.36.211.159:1 and 217.36.211.159:2



The VNC sessions are close to real-time on a good connection. Our UK to UK testing had delays of less than 1 second.

You will be presented with an authentication dialog box. Enter the password you were given by AMS with your time slot.



Once you are logged in, the background text will indicate which robot you are currently logged in to. This will be either “Front GTR” to indicate the forward facing robot, or “Rear GTR” to indicate the reverse facing robot.

STEPS FOR COMPETITORS

From here you will want to open the terminal (clicking on the black screen at the top highlighted below).

Be aware that the VNC sessions are not capable of accepting copy/paste text or files. They are also unable to display the robot camera view so trying to do so will cause the scripts to fail.

If you need at any point a password for the Pi login itself, these are:

User: pi

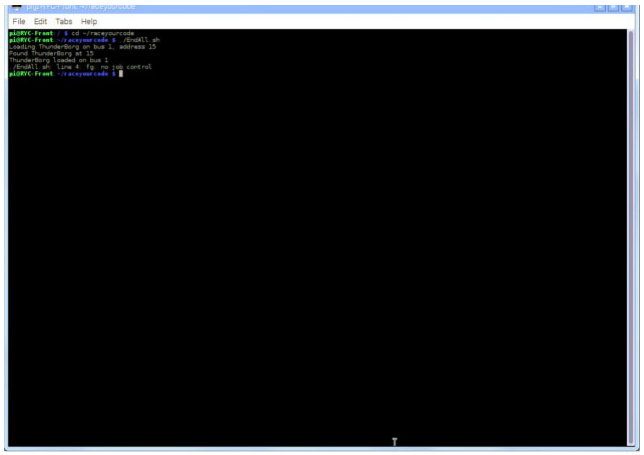
Password: raspberry

This user has sudo permissions so that no root password should be necessary.

You can get a root terminal by opening the standard terminal and running: sudo bash

Next, change to the raceyourcode directory, by typing in cd ~/raceyourcode

Execute ./EndAll.sh to kill any running python script and stop the motors (if they are running). You will want to do this on both robots.



From here you can execute scripts. We recommend you put any code for the robot into the ~/raceyourcode directory during your slot as this is where the robot will look for code (RaceYourCode_Front.py and RaceYourCode_Rear.py) during your timed three lap run.

If you want to kill a script you are running use the Ctrl+C. If your script is unresponsive, open another terminal window and execute the ./EndAll.sh (from the ~/raceyourcode directory)

Emulate the start light sequence

For your 3 lap run the start lights will be cycled. However, in your initial 45 minutes of run time, the start lights will remain off. To execute your code as if the start lights had cycled, you need to:

1. Start the rear robot normally by executing the sudo ./RaceYourCode_Rear.py (on the rear robot)
2. Wait until the rear robot is outputting FPS values to the terminal
3. On the front robot, execute sudo ./NoLights_Front.py
4. Both robots should start moving on their own after a short delay. The front robot will behave as though it has seen the start lights, causing it to start driving and to send the starting command to the rear robot at the same time.

STEPS FOR COMPETITORS

Towards the end of your 45 minutes

You will get a warning via twitch that your time is coming to a close. We recommend immediately running the ./EndAll.sh on both robots, and saving any work that you have open. We also recommend running a sudo sync command to ensure all changes are written to the sdcard. Best practice is to then issue the sudo halt on both robots.

During this testing time you may ask for any of the following: (These could take a few minutes for us to perform)

1. The robots to be placed back at their starting positions
2. A reset of the track timer
3. A power cycle of one or both robots
4. We can log in to the robots and assist with error messages if needed

For the 3 lap race

At the end of the 45 minutes, we will switch off both robots, reset them in the correct starting positions on the track and power them on. We highly recommend NOT logging in to either robot during the three lap race.

Once the two minutes has elapsed, we will run through the light sequence and your timed laps will commence.

The lap timer is used for indicative purposes. As the rear robot will cross shortly after the first, it will indicate a very short amount of time between the first and second. The judged time will be the time taken for both robots to complete at least three laps. There is a maximum time limit of 15 minutes. If three laps are not achieved by both robots in the time limit the judged time will be the full 15 minutes.

Programming the robot to automatically stop after three laps is not recommended due to detection inconsistencies with the start marker. Instead we recommend that your timed run is programmed to run indefinitely.

During your 3 lap run we will NOT login to the robots and assist you or manually intervene with any stuck robots. You are allowed to log in to either robot during this time to help them if you need to.

If you crash during the 45 minute run

If you crash, stop the robots by executing the EndAll.sh shell script on both robots.

You can then navigate the front robot using the ./ManualMotors_Front.py script and navigate on the rear robot using the ./ManualMotors_Rear.py scripts.

1. Remember to run each script on the appropriate VNC window (RYC-front for _Front, RYC-rear for_Rear).
2. Remember that there is approximately a 10 second delay between what you execute in the robot and what you will see on the twitch stream.
3. If you cannot see a robot, you can ask us to change the camera view via the Twitch chat window.

Some common problems

If you get a camera error on either robot it may be best to restart the robot using sudo reboot

Wait a minute or so then try and re-connect.

It's possible to disconnect the battery and restart the Pi, especially if there has been a hard crash. If this happens, please tell us through the chat window and we can investigate. Extra time may or may not be awarded. This is solely at the discretion of the stewards. Due to this, it is advisable to save any changes regularly.

It can be difficult to stop the robots quickly if something goes wrong. A helpful hint is to run two terminals while testing, one for running the code, and a second for running EndAll.sh / editing scripts.

STEPS FOR COMPETITORS

The Twitch stream and chat can sometimes get stuck. If things do not appear to be working it can usually be fixed by refreshing the browser page.

The robots may prove difficult to control at slow speeds, in particular steering control is poor at very low power (< 30 %) so it is best to keep power settings above about 50%. Additionally at very slow speeds the stuck detection may incorrectly think the robot is failing to move!

Some common error messages

- socket.error: [Errno 98] Address already in use or *** Error in `python': double free or corruption
  Either of these messages usually mean that there is already code running (both robots boot up with the standard code running). You can fix the problem by running the ./EndAll.sh script from the ~/raceyourcode directory.
- Command not found

There are several possible causes:

1. Not in the right directory - you may need to run cd ~/raceyourcode firstNot in the right directory - you may need to run cd ~/raceyourcode first
2. Extra space - Twitch chat inserts an extra space sometimes, for example ./ EndAll.sh should be ./EndAll.sh instead
3. Wrong permissions - try running chmod +x *.py *.sh and see if the script now runs. This is especially common if you are copying files from a Windows machine
4. Spelling errors - Double check the name of the script, Linux cares about case! e.g. raceyourcode_front.py will fail, it should be sudo ./RaceYourCode_Front.py instead
   sudo ./raceyourcode_front.py will fail, it should be sudo ./RaceYourCode_Front.py instead

- Client is not authorized to connect to Server or Gtk-WARNING **: cannot open display: :1.0
  This points to the script trying to show an image while running, which it cannot do via the VNC connection. Look for the "Image processing debugging settings" section in the script you are trying to run and make sure these values are set to False:
  ImageProcessor.showProcessing and ImageProcessor.predatorView

- socket.error: [Errno 12] Permission denied
  Our scripts open network connections to send their telemetry data, this typically needs extra permissions to run. Run the same script using sudo instead. For example:
  sudo ./ RaceYourCode_Front.py will work./ RaceYourCode_Front.py will fail

- : No such file or directory
  This error is usually caused by Windows style line endings. This can be fixed by installing dos2unix using the command sudo apt-get install dos2unix and then using the command on all of the scripts like this: dos2unix *.py *.sh
  It should be safe to run the command on any of the scripts, even if they do not have the problem

Getting help

The fastest way to get help whilst you are running is to comment in the Twitch chat window. To do this, you will need a twitch account. Twitch requires an email address for commenting, so we recommend that you sign up for Twitch well before your race begins.

We don't anticipate it, but should there be a problem with too many comments or spam comments, it is possible to ignore people in the chat window. You can also report this as spam.

GETTING THE CODE

Now that we know how to log in and get scripts running, let's take a look at the standard code and some useful scripts we can use during our 45 minute run.

Downloading code on Windows:

Download file from https://www.formulapi.com/raceyourcode.zip

You should be able to use your standard ZIP file program to unzip the contents

The password for the file is: access-granted

Downloading code on Linux from a terminal:

```
sudo apt-get install unzip
cd ~
wget https://www.formulapi.com/raceyourcode.zip
unzip raceyourcode.zip
```

The password is: **access-granted**

```
cd raceyourcode
chmod -R +x *.py *.sh
```

The code is licensed for your use with a non-open source license, check the LICENSE.txt file for the full license terms.

CHANGING CODE

You may wish to alter code before your run. For example, code you have tested in the simulator. The Raspberry Pi will have Internet access during your slot, so you can make use of this to transfer your code over. For example:

For example:

- You could use the wget command to download files directly from the internet
- You could browse the web using the built-in web browser to access files
- You could put your code on a site like GitHub and check it out on the robot

Please note that the race code is NOT open-source, so if you use a public service like GitHub we ask that you remove the files after your run has finished.

Why? See https://www.formulapi.com/comment/240#comment-240

You may use apt-get and similar commands to install programs and libraries, but given the short amount of time we recommend you try and keep this to small downloads only or avoid it all together.

You will not be able to access your code once your time has expired. Code changes made during a race will be overwritten after the race for subsequent users.

THE STANDARD CODE

Our standard code is made up of a number of files, most of the code is shared by both robots. Scripts ending in Front are intended for the leading robot, those ending in Rear are intended for the trailing robot.

There are more detailed explanations of the scripts and their parts in the Guides folder provided with the code. You should be given access details to all of the code for experimenting and development as part of your registration.

Some scripts need to be run with sudo to work as they need permission to send network messages.

- RaceYourCode_Front.py / RaceYourCode_Rear.py
  These scripts run the robots for the actual race, but they do not do much of the work. The front script will watch for our start lights, then communicate this information to the rear robot which cannot see the lights itself. These scripts are auto-started when the robot powers up.

- NoLights_Front.py
  This is an alternative to the main RaceYourCode_Front.py script which can be used to start the robots racing without waiting for the lights. The trailing robot will be told the lights sequence has gone to GO so you will want to run RaceYourCode_Rear.py on the trailing robot at the same time.

- ManualMotors_Front.py / ManualMotors_Rear.py
  These scripts provide a quick way to manually move the robots during your allotted testing time. They are not meant for proper driving, more for positioning robots back on the track if they get stuck..

- EndAll.sh
  Just a quick script which will stop any of the standard scripts which are running and stop the robot as well. We recommend you run this at the start of your allotted test session so you can choose which scripts you want to run manually and see their output.

- SimulationFront.py / SimulationRear.py / SimulationImages.py
  These are explained in more detail later on, but these are scripts to help with development of the code without the robots, allowing you to run the bulk of the code using the provided simulator to generate camera data and track position.

- Settings.py / Globals.py
  These scripts hold settings and defaults respectively for the standard processing code. Most of the settings can be changed on the fly by overwriting Settings.py while the scripts are in control, but not all of them. The values in both of these files are shared between all of the scripts.

- ImageProcessor.py
  This is where the bulk of the code lives. This file is responsible for both the camera processing and the calculations for the motor output speeds. The code in this file is heavily maths and OpenCV based, but has been kept fairly simple. If you wish to alter the image processing behavior you will likely want to modify this script.

- Race.py
  This script runs alongside the image processing and is mostly intended for determining target track positions and other race type decisions. There are a number of examples provided for how this thread could be used, but it can access and alter most values in the image processing code as well.

- RaceCodeFunctions.py
  This script provides the simplified race API used in our Race.py examples. They provide functions such as logging and feedback from the processing code. A full list of the functions can be found here: https://www.formulapi.com/race-code/functions

- ThunderBorg.py
  This script provides a simple API to control the on-board ThunderBorg motor controller without the need to write I2C messages by hand.

IMPORTANT NOTE:
Most of the code is common with our Formula Pi Monster series code. As such there is a lot of useful information about the race code which may prove useful here: https://www.formulapi.com/race-code and also some insight into how our logic works here: https://www.formulapi.com/tags/image-processing

RUNNING IN SIMULATION

In order to test and develop the code without access to the real robots and track we have provided a Java based simulator which tracks the robots' movement and generates an image stream to act as the camera input.

The simulator does have some limitations compared to the real robots:

- Everything is evenly lit, there are no shadows or bright spots
- Outside of the track is purely grey, in reality there are a lot of things in the background
- Real world camera data is noisy and has a lot of motion blur, the simulation camera is nearly perfect
- The start lights do not project light like the real ones
- It can only handle one robot at a time, not both
- There is no collision detection, the robots can run straight though the track walls
- The robots are only roughly modelled, they do not behave exactly like the real robots
- The robots do not perform as well at low speeds, the simulation works better than it should in this case
- The simulation has no collision detection for the walls, if your robot touches the black part of the track in the simulation it will likely run into a wall on the real track
- There is some level of camera lag simulated, but it is only a rough approximation of the robot behaviour
- The robot size in the simulation is smaller than the real robots so care needs to be taken when cutting corners closely
- The rear robot will not start at the correct position in the simulation. In the real race, the rear robot starts about 1m behind the front
- Having said all that, it is possible to use the simulator to improve the code. We have even used it ourselves to tune updates for new robot models before testing them on the real track.

Running the simulator

The simulation comes in two parts: the simulator itself and the simulation mode Python script. While both can be run on a single machine our usual recommendations are:

1. Run the simulator on a Windows machine running a recent version of Java. This needs to be a machine with graphics capable of at least OpenGL 3.2. It should run on a Linux machine as well, Macs may work but are unsupported.
2. Run the simulation mode Python script on a Raspberry Pi 3 connected via Ethernet.

We recommend a Pi 3 so that you can confirm the real robot can keep up with everything your code tries to do. It should run on any platform where you can install Python 2.X (at least 2.7) and the OpenCV library for Python (we will run with 2.4.13). We recommend Ethernet or a very reliable Wi-Fi connection to keep the image stream lag free and consistent.

If you are running both halves on the same machine the scripts should be ready to go straight away. If not you will need to edit SimulationFront.py and SimulationRear.py so that the simulationIP value towards the top is the IP address of the machine running the Java half of the simulation.

Actually running the simulator is a fairly simple process, but there are quite a few steps:

3. Load the Java based simulation (Simulation.bat on Windows).
4. Press the Off button under AI Control, only one robot should be shown on the track at this point.
5. Press the Start button under Video Stream Settings.
6. Make sure the Off button is greyed out under Start Settings, if not press it.
7. Press Reset under Bot Simulation to place the MonsterBorg into the correct lane.

RUNNING IN SIMULATION

8. Run the SimulationFront.py script on the Raspberry Pi and wait until it has printed "WaitForGo()". After this it will start printing FPS readings on the default settings, they should be ~ 30.0. You will probably want to wait for about 10 seconds at this point for the script to be ready.

9. Press the Green button under StartSettings. The LED on the MonsterBorg diagram should turn on, if not it failed to see the lights. The script should print "Lights: 1 - Green".

10. Press the Red button under StartSettings. The script should print "Lights: 2 - Red".

11. Press the Green button under StartSettings again. The LED on the MonsterBorg diagram should turn off, if not it failed to see the lights. The script should print "Lights: 3 - Green". This will be followed by "Lights: GO". The MonsterBorg will begin the race at this point.

12. Finally press the Off button under StartSettings as soon as possible. With the green lights still on the script can get confused and reduce the green level in the image too much.

Running the simulation for the trailing robot instead is shorter is it will ignore the lights:

13. Load the Java based simulation (Simulation.bat on Windows).

14. Press the Start button under Video Stream Settings.

15. Press Reset under Bot Simulation to place the MonsterBorg into the correct lane.

16. Run the SimulationRear.py script on the Raspberry Pi and wait for it to start rotating the robot.

17. Once the MonsterBorg has rotated itself it should start racing automatically.

See the detailed guides for more information on what can be done with the simulation scripts.

Testing with real images

If you re altering the image processing the simulator may not be accurate enough to real images. You can use the SimulationImage.py script to run through a directory of JPEG images and show you what the results from the image processing are and allow a means to debug real-world camera problems.

For this you will want some real images from a running robot. A suitable set can be found on our website here: https://www.formulapi.com/blog/monster-raw-footage-analysis

The same page explains the various differences between these images and the image data from the simulator. The image set includes a set of light changes, clean laps, messed up laps, crashes, and even a short stint flipped upside-down. The images were taken at 30 fps, but there are some missing sections in the image set as explained on the page.

Additional information

For further information, please check the "Guides" folder included in the standard code zip. Some text files in this folder give insight into modifying particular scripts.

The "Blog" tab on FormulaPi.com website will also have relevant information. There is also a raceyourcode thread on the FormulaPi forum.