RaceTrack 1.0 Documentation and User Guide

Tom Smith, December 28 2020

About

RaceTrack is an app written in Python and compiled for use on Windows 10 machines that's designed to facilitate communication between a smartphone app and Google Earth Pro, for the purpose of plotting the position of two smartphones on Google Earth in semi-real time, intended for use during the Riverbank Run. It was build in Spyder via Anaconda, and can be rewritten in it, or recompiled using pyinstaller from the Anaconda cmd prompt, assuming the packages it requires are installed (which can be done from the Anaconda cmd prompt as well). This guide will cover required components, setup, operation, and troubleshooting, and includes the app's source code at the end.

Required Components

1. Two smartphones running the full version of the app GPS2IP.
2. Google Earth Pro (the desktop app).
3. The app itself to be open and running the same computer running Google Earth Pro.
4. Two open UDP ports on the network of the computer that's running RaceTrack.

Smartphones and GPS2IP

HTTPS://APPS.APPLE.COM/US/APP/GPS-2-IP/ID408625926

GPS2IP is a smartphone app that sends live GPS data to a specified UDP port on a specified network. This is where RaceTrack gets its data. The app is a free download, but in order for it to work with RaceTrack, the full version must be purchased ($6.99). Without the full version, it will not send GPS data while the app is not in focus or the phone is locked, and can only send a limited number of messages.

GPS2IP Configuration

TIME BETWEEN SENDING POSITIONS
- No Delay

GPS ACCURACY
- The very best

NMEA MESSAGES TO SEND
- GGA

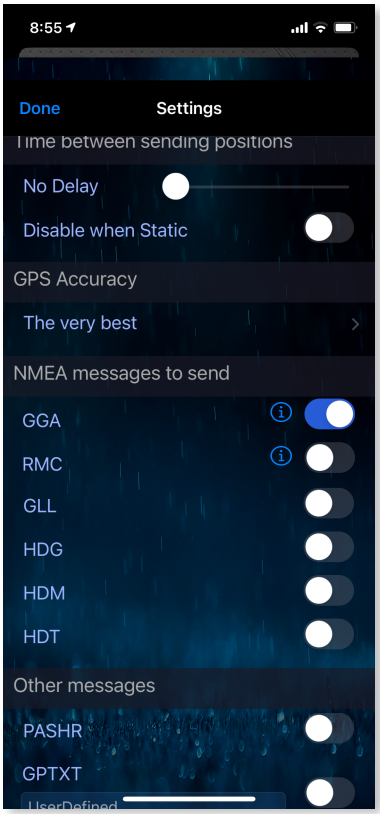
GENERAL SETTINGS
- Operate in background mode

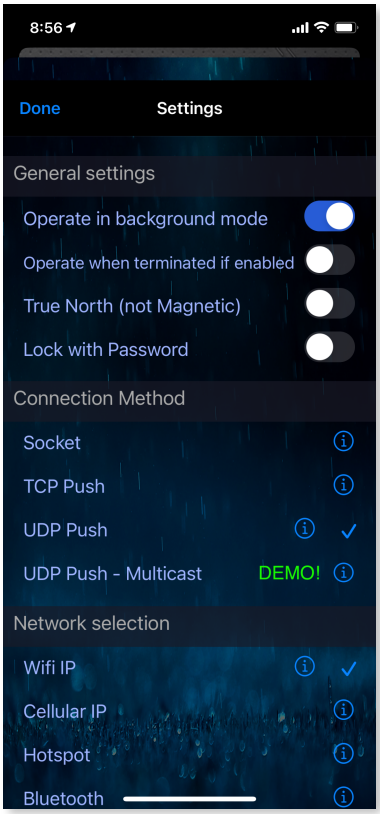
CONNECTION METHOD
- UDP Push

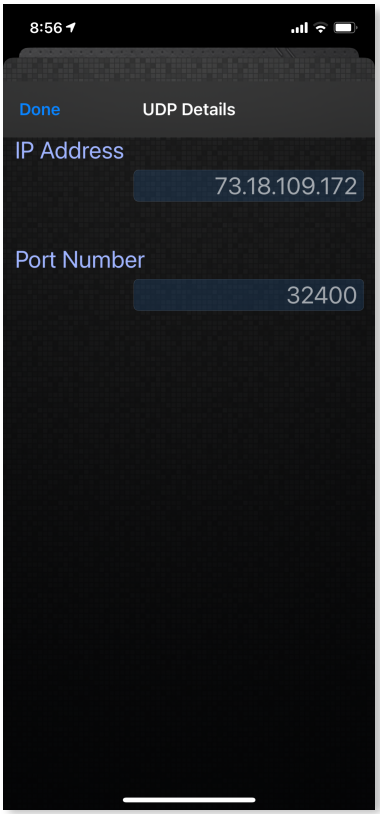
UDP DETAILS
- Tap the “i” next to UDP push to open.
- IP Address should be set to the PUBLIC IP of the machine that’s running RaceTrack.
- Port Number should be set to the men’s or women’s UDP port that’s assigned in RaceTrack. I.E. if you’re configuring the phone for the men’s race, and the men’s race has port 9091 assigned in RaceTrack, port 9091 should be entered as the port on that phone.

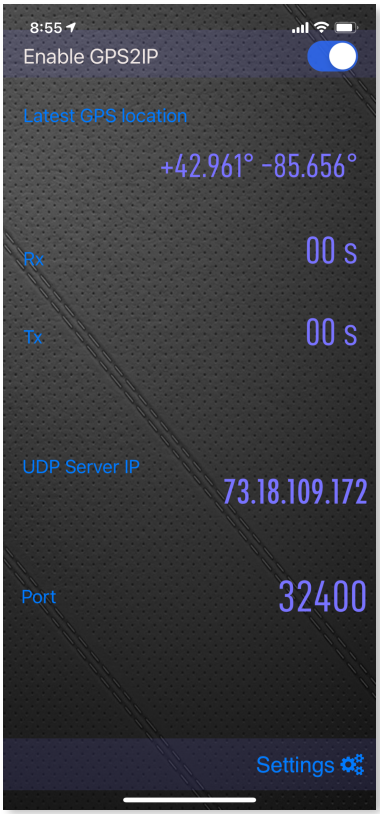
NOTES:
Once all the settings are entered, you can turn on the “Enable GPS2IP” toggle at the top of the main page to begin sending data. Optionally, you may wish to enable “Operate when terminated if enabled” and “Lock with Password” in General Settings for added peace of mind. All other settings should be followed exactly as presented here.

Data and battery consumption seem to be minimal, but it might be a good idea to issue the operator a power bank just in case.

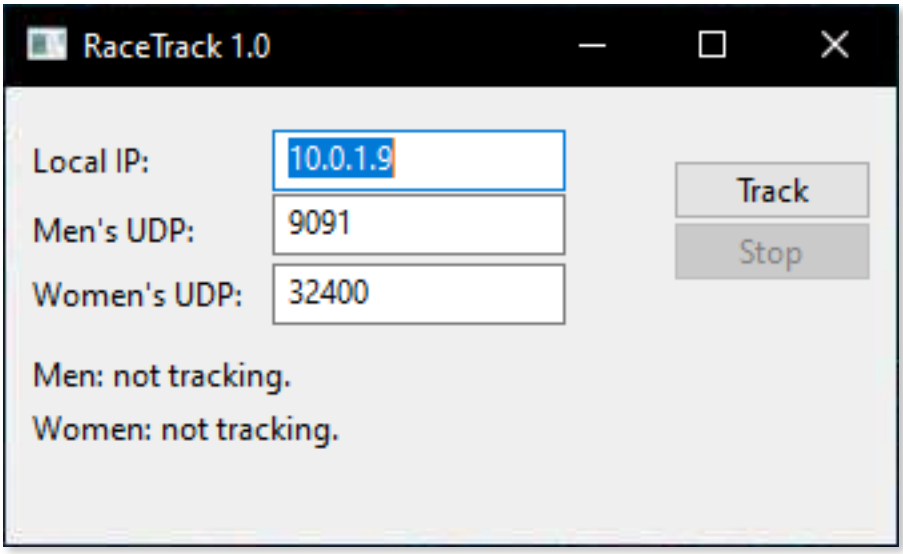




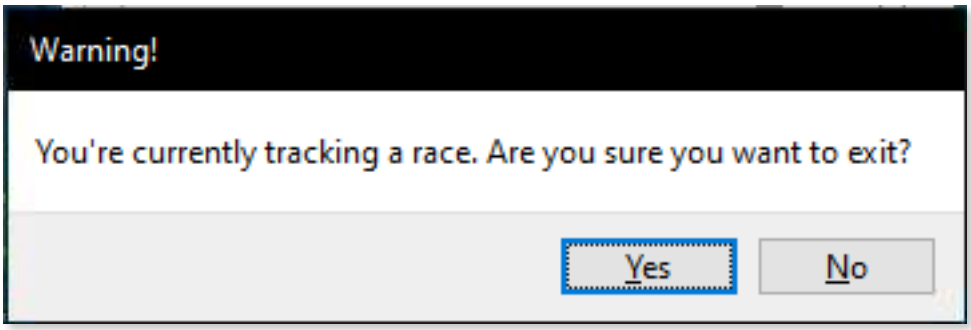




RaceTrack Configuration and Operation



RaceTrack is very easy to configure and operate. When launched, it will automatically populate the machine's Local IP text box, while the Men's UDP and Women's UDP will be assigned to 9091 and 32400 by default as these were the ports I used when building the app on my home network. Fill in the appropriate UDP port numbers and press "Track" to begin tracking. As data flows in, the longitude and latitude of each device will be displayed below the boxes and change as the data changes.



"Stop" should always be pressed before closing the app. This ensures the connections to the UDP sockets are closed before the app quits, so that if the app needs to be restarted for some reason, it'll be able to connect to those sockets again (otherwise, a reboot of the computer may be required). As a failsafe, if you try to exit the program while still tracking, you'll be presented with a warning dialogue first. If you choose yes, it will try to close the socket connections before killing the application.

At launch, RaceTrack will generate two files in the same directory it was launched from: *MenLiveTrackLoad.kml* and *WomenLiveTrackLoad.kml*. When you begin tracking, it will generate two more files in that directory: *mensposition.kml* and *womensposition.kml*. The *LiveTrackLoad* files are the ones which need to be opened in Google Earth. These files

reference the *position* files, and refreshes those files every second. Likewise, RaceTrack overwrites the *position* kml files every second, which is what allows the semi-real time placement of the markers on Google Earth.

All four KML files reference file locations; the paths will be generated automatically by RaceTrack, but because of this, these six files need to be located in the same directory that RaceTrack was launched from:

1. MenLiveTrackLoad.kml
2. WomenLiveTrackLoad.kml
3. mensposition.kml
4. womensposition.kml
5. Marker_M.png
6. Marker_W.png

LiveTrackLoad files contain references to position files. Position files contain references to the marker files. The best practice is to launch and run RaceTrack from the folder that it was distributed in.

KML Files

The source code for the KML files is listed here in the event that one needs to be remade for some reason. The file names are case sensitive. Copy and paste them into Notepad and save them as .kml files. Don't worry about file paths or GPS coordinates in these; RaceTrack will overwrite them appropriately.

MensLiveTrackLoad.kml

```
<?xml version='1.0' encoding='UTF-8'?><kml xmlns='http://
www.opengis.net/kml/2.2' xmlns:gx='http://www.google.com/kml/ext/
2.2'><Document id='8655'><NetworkLink id='8656'><name>Network Link</
name><Link id='8657'><href>C:\Users\Tom\Documents\Python
Programming\RaceTrack\menspositon.kml</href><refreshMode>onInterval</
refreshMode><refreshInterval>1</refreshInterval></Link></
NetworkLink></Document><NetworkLinkControl><minRefreshPeriod>1</
minRefreshPeriod></NetworkLinkControl></kml>
```

WomensLiveTrackLoad.kml

```
<?xml version='1.0' encoding='UTF-8'?><kml xmlns='http://
www.opengis.net/kml/2.2' xmlns:gx='http://www.google.com/kml/ext/
2.2'><Document id='8655'><NetworkLink id='8656'><name>Network Link</
name><Link id='8657'><href>C:\Users\Tom\Documents\Python
Programming\RaceTrack\womensposition.kml</
href><refreshMode>onInterval</refreshMode><refreshInterval>1</
refreshInterval></Link></NetworkLink></
Document><NetworkLinkControl><minRefreshPeriod>1</minRefreshPeriod></
NetworkLinkControl></kml>
```

mensposition.kml

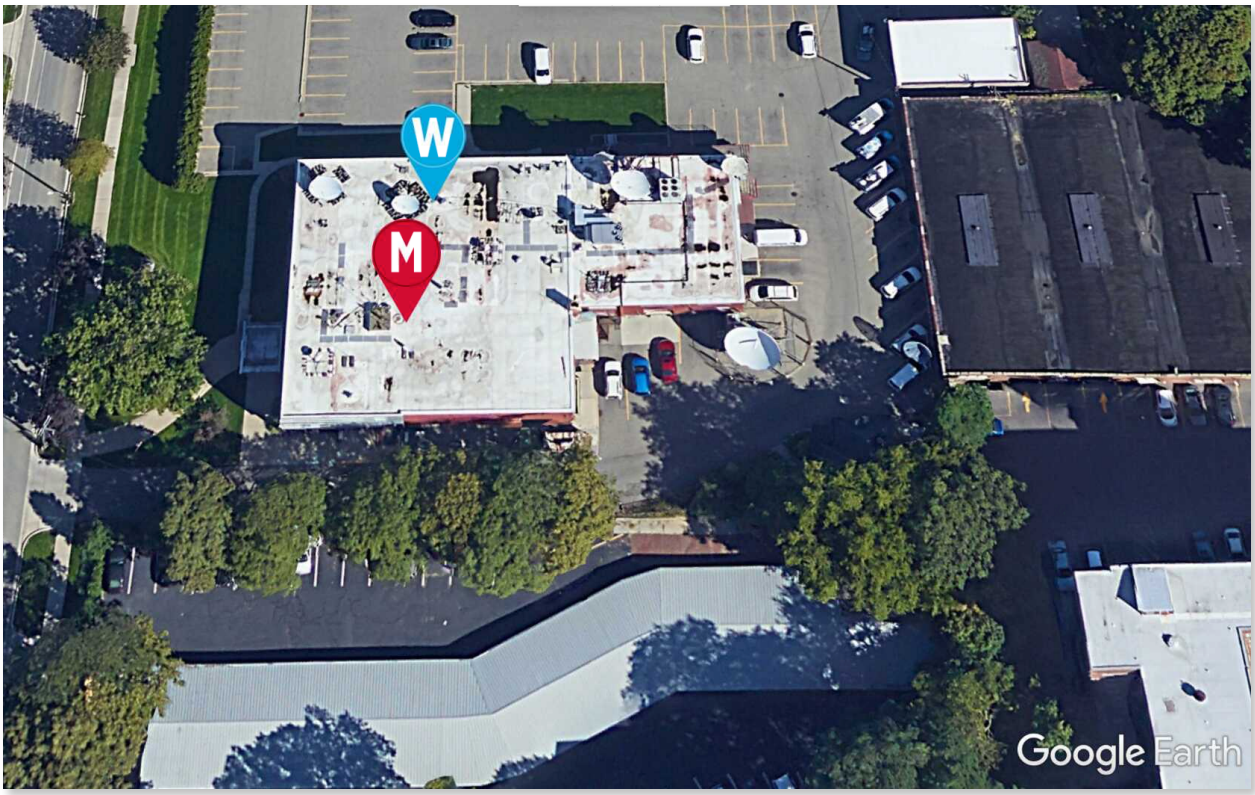
```
<?xml version='1.0' encoding='UTF-8'?><kml xmlns='http://
www.opengis.net/kml/2.2' xmlns:gx='http://www.google.com/kml/ext/
2.2'><Document id='6076'><Style id='6082'><IconStyle
id='6083'><colorMode>normal</colorMode><scale>5</scale><heading>0</
heading><Icon id='6084'><href>C:\Users\Tom\Documents\Python
Programming\RaceTrack\Marker_M.png</href></Icon></IconStyle></
Style><Placemark id='6081'><styleUrl>#6082</styleUrl><Point
id='6080'><coordinates>-85.65624883333334,42.96042116666667,0.0</
coordinates></Point></Placemark></
Document><NetworkLinkControl><minRefreshPeriod>1</minRefreshPeriod></
NetworkLinkControl></kml>
```

womensposition.kml

```
<?xml version='1.0' encoding='UTF-8'?><kml xmlns='http://
www.opengis.net/kml/2.2' xmlns:gx='http://www.google.com/kml/ext/
2.2'><Document id='6076'><Style id='6082'><IconStyle
id='6083'><colorMode>normal</colorMode><scale>5</scale><heading>0</
heading><Icon id='6084'><href>C:\Users\Tom\Documents\Python
Programming\RaceTrack\Marker_W.png</href></Icon></IconStyle></
Style><Placemark id='6081'><styleUrl>#6082</styleUrl><Point
id='6080'><coordinates>-85.65621933333334,42.96055333333334,0.0</
coordinates></Point></Placemark></
Document><NetworkLinkControl><minRefreshPeriod>1</minRefreshPeriod></
NetworkLinkControl></kml>
```

Google Earth Pro Configuration

HTTPS://WWW.GOOGLE.COM/EARTH/VERSIONS/#EARTH-PRO



Google Earth Pro can be downloaded for free from the URL above. Open the *LiveTrackLoad* files from the file menu and the markers should appear at their most recent location. In the sidebar open the drop down menu for the files and look for the entry labeled “Network Link”; if it’s green, it sees the *position* file and is working correctly. If it’s red, something is broken with the path to the *position* file in the *LiveTrackLoad* file. Also in the sidebar, I suggest turning off all layers except for Terrain for the cleanest look. 3D buildings especially will sometimes cover up or clip the makers. In the menu bar, select tools and disable the sidebar, toolbar, set Show Navigation to never, and enter fullscreen. Running this on an INET computer will give us access to it in Ignite, allowing us to put it on air.

We could leave the window set to give us a full view of the course, or in theory, someone at the computer could act as a Google Earth camera operator, zooming in and out on specific markers and locations as requested.

Source Code

```
# -*- coding: utf-8 -*-
"""
RaceTrack 1.0
Created on Thu Dec 17 08:11:09 2020

@author: Tom Smith
"""

import time, wx, pynmea2, socket, threading, os

"""
Time allows pausing of threads to avoid crashing.
wx builds the GUI.
pynmea2 decodes the NMEA data send from GPS2IP into standard long/lat decimal notation.
socket opens/listens to/gets data from UDP port sockets.
threading enables threading, allowing the infinite tracking loops to run simultaneously and not
crash the GUI.
os gets the path of the directory the app was launched in so the file paths are written correctly
in the KML files.
"""

hostname = socket.gethostname() #Gets the machine's friendly name.
local_ip = socket.gethostbyname(hostname) #Getting local IP for the app from friendly name.


app_directory = os.getcwd() #Getting the current directory path.
mens_live_str  =  "<?xml  version='1.0'  encoding='UTF-8'?><kml  xmlns='http://www.opengis.net/kml/
2.2'         xmlns:gx='http://www.google.com/kml/ext/2.2'><Document         id='8655'><NetworkLink
id='8656'><name>Network           Link</name><Link           id='8657'><href>{}\menspositon.kml</
href><refreshMode>onInterval</refreshMode><refreshInterval>1</refreshInterval></Link></
NetworkLink></Document><NetworkLinkControl><minRefreshPeriod>1</minRefreshPeriod></
NetworkLinkControl></kml>".format(app_directory)
mens_live_file = open("MenLiveTrackLoad.kml", "w")
mens_live_file.writelines(mens_live_str)
mens_live_file.close() #Generating MenLiveTrackLoad.kml
womens_live_str  =  "<?xml  version='1.0'  encoding='UTF-8'?><kml  xmlns='http://www.opengis.net/kml/
2.2'         xmlns:gx='http://www.google.com/kml/ext/2.2'><Document         id='8655'><NetworkLink
id='8656'><name>Network          Link</name><Link          id='8657'><href>{}\womensposition.kml</
href><refreshMode>onInterval</refreshMode><refreshInterval>1</refreshInterval></Link></
NetworkLink></Document><NetworkLinkControl><minRefreshPeriod>1</minRefreshPeriod></
NetworkLinkControl></kml>".format(app_directory)
womens_live_file = open("WomenLiveTrackLoad.kml", "w")
womens_live_file.writelines(womens_live_str)
womens_live_file.close()
prevent_close = False #Generating WomenLiveTrackLoad.kml

#Frame the app is displayed in. Also enables the closing warning.
class MyFrame(wx.Frame):
    def __init__(self, parent, title):
        super(MyFrame, self).__init__(parent, title=title, size=(350,210))

        self.panel = MyPanel(self)
        self.Bind(wx.EVT_CLOSE, self.onClose) #Presents warning if closing while tracking.

    def onClose(self, event): #The warning itself.
        global prevent_close
        if prevent_close:
            dlg = wx.MessageDialog(self, "You're currently tracking a race. Are you sure you want
to exit?", "Warning!", wx.YES_NO | wx.ICON_QUESTION)
            if dlg.ShowModal() == wx.ID_YES:
                self.Destroy()  # frame
```

```
                dlg.Destroy()
                MyPanel.close_ports(self.panel, event)
        if not prevent_close:
            self.Destroy()



class MyPanel(wx.Panel):
    def __init__(self, parent):
        super(MyPanel, self).__init__(parent)
        self.root_directory = os.getcwd() #Getting the directory.
        self.local_ip_box = wx.TextCtrl(self, wx.ID_ANY, local_ip, pos = (100, 16))
        self.local_ip_label = wx.StaticText(self, label = "Local IP:", pos = (10,20))
        self.mens_udp_box = wx.TextCtrl(self, wx.ID_ANY, "9091", pos = (100, 40))
        self.mens_udp_label = wx.StaticText(self, label = "Men's UDP:", pos = (10, 46))
        self.womens_udp_box = wx.TextCtrl(self, wx.ID_ANY, "32400", pos = (100, 66))
        self.womens_udp_label = wx.StaticText(self, label = "Women's UDP:", pos = (10, 70))
        self.start_btn = wx.Button(self, label = "Track", pos = (250, 27))
        self.start_btn.Bind(wx.EVT_BUTTON, self.trackingcond)
        self.end_btn = wx.Button(self, label = "Stop", pos = (250,50))
        self.end_btn.Bind(wx.EVT_BUTTON, self.trackingcond)
        self.end_btn.Enable(False)
        self.tracking = False #Condition that triggers the loops when true.
        self.men_pos_label = wx.StaticText(self, label = "Men: not tracking.", pos= (10, 100))
          self.women_pos_label = wx.StaticText(self, label = "Women: not tracking.", pos = (10,
120))


    def open_ports(self, event): #Must run to open ports in app for data.
        self.UDP_IP1 = self.local_ip_box.GetValue()
        self.UDP_PORT1 = int(self.mens_udp_box.GetValue())
        self.UDP_IP2 = self.local_ip_box.GetValue()
        self.UDP_PORT2 = int(self.womens_udp_box.GetValue())
        self.sock1 = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        self.sock2 = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        self.sock1.bind((self.UDP_IP1, self.UDP_PORT1))
        self.sock2.bind((self.UDP_IP2, self.UDP_PORT2))

    def close_ports(self, event): #Closes ports when tracking stops so they can be opened again
in the same instance.
        if self.sock1:
            self.sock1.shutdown(1)
            self.sock1.close()
        if self.sock2:
            self.sock2.shutdown(1)
            self.sock2.close()

    def track_men(self, condition): #Gets data from men's UDP and distributes it appropriately.
        while condition:
            if self.sock1.recv is not None:
                men_data, men_addr = self.sock1.recvfrom(1024)
                men_decode = men_data.decode('latin-1') #Decoding from bytes to text.
        #Gets NMEA GPS coordinates and splits them to vars.
                men_lat,men_dir1,men_long,men_dir2 = men_decode.split(",")[2:6]
        #Builds an nmeaobj to parse data from that string.
                nmeaobj_men = pynmea2.parse(men_decode)
        #Converting from $GPGGA to more usable decimal notation
                lat_men = nmeaobj_men.latitude
                long_men = nmeaobj_men.longitude
                lat_men_str = str(lat_men)
                long_men_str = str(long_men)
                men_output_str = "Men: {},{}".format(long_men_str,lat_men_str)
                print(men_output_str)
```

```
                    men_kml_str = "<?xml version='1.0' encoding='UTF-8'?><kml xmlns='http://
www.opengis.net/kml/2.2'  xmlns:gx='http://www.google.com/kml/ext/2.2'><Document  id='6076'><Style
id='6082'><IconStyle          id='6083'><colorMode>normal</colorMode><scale>5</scale><heading>0</
heading><Icon         id='6084'><href>{}\Marker_M.png</href></Icon></IconStyle></Style><Placemark
id='6081'><styleUrl>#6082</styleUrl><Point       id='6080'><coordinates>{},{},0.0</coordinates></
Point></Placemark></Document><NetworkLinkControl><minRefreshPeriod>1</minRefreshPeriod></
NetworkLinkControl></kml>".format(self.root_directory,long_men_str,lat_men_str)
                men_kml = open("menspositon.kml", "w")
                men_kml.writelines(men_kml_str)
                men_kml.close()
                self.men_pos_label.SetLabel(men_output_str)
                time.sleep(1)
                if self.labelName == 'Stop':
                    self.men_pos_label.SetLabel("Men: not tracking.")
                    print ("Stopping men's tracking.")
                    break

     def track_women(self, condition): #Exactly the same as the men track but with different
variable names and UDP port.
        while condition:
            if self.sock2.recv is not None:
                women_data, women_addr = self.sock2.recvfrom(1024)
                women_decode = women_data.decode('latin-1)')
        #Gets NMEA GPS coordinates and splits them to vars
                women_lat,women_dir1,women_long,women_dir2 = women_decode.split(",")[2:6]
        #Builds an nmeaobj to parse data from that string
                nmeaobj_women = pynmea2.parse(women_decode)
        #Converting from $GPGGA to more usable decimal notation
                lat_women = nmeaobj_women.latitude
                long_women = nmeaobj_women.longitude
                lat_women_str = str(lat_women)
                long_women_str = str(long_women)
                #lat_women_float = float(lat_women)
                #long_women_float = float(long_women)
                women_output_str = "Women: {},{}".format(long_women_str,lat_women_str)
                print(women_output_str)
                     women_kml_str = "<?xml version='1.0' encoding='UTF-8'?><kml xmlns='http://
www.opengis.net/kml/2.2'  xmlns:gx='http://www.google.com/kml/ext/2.2'><Document  id='6076'><Style
id='6082'><IconStyle          id='6083'><colorMode>normal</colorMode><scale>5</scale><heading>0</
heading><Icon          id='6084'><href>{}\Marker_W.png</href></Icon></IconStyle></Style><Placemark
id='6081'><styleUrl>#6082</styleUrl><Point       id='6080'><coordinates>{},{},0.0</coordinates></
Point></Placemark></Document><NetworkLinkControl><minRefreshPeriod>1</minRefreshPeriod></
NetworkLinkControl></kml>".format(self.root_directory,long_women_str,lat_women_str)
                women_kml = open("womensposition.kml", "w")
                women_kml.writelines(women_kml_str)
                women_kml.close()
                self.women_pos_label.SetLabel(women_output_str)
                time.sleep(1)
                if self.labelName == 'Stop':
                    self.women_pos_label.SetLabel("Women: not tracking.")
                    print("Stopping women's tracking.")
                    break

    def trackingcond(self, event): #Tied to buttons, starts and stops tracking threads/loops.
        global prevent_close
        button = event.GetEventObject()
        self.labelName = button.GetLabel()
        if self.labelName == 'Track': #Starts tracking.
            prevent_close = True
            self.tracking = True
            self.start_btn.Enable(False)
            self.local_ip_box.Enable(False)
            self.mens_udp_box.Enable(False)
            self.womens_udp_box.Enable(False)
```

```
            self.end_btn.Enable(True)
            self.open_ports(event)
              threading.Thread(target=self.track_men, args=(self.tracking,), daemon=True).start()
#Opens thread to track men.
             threading.Thread(target=self.track_women, args=(self.tracking,), daemon=True).start()
#Opens thread to track women.
        else: #Stops tracking.
            prevent_close = False
            self.tracking = False
            self.close_ports(event)
            self.local_ip_box.Enable(True)
            self.mens_udp_box.Enable(True)
            self.womens_udp_box.Enable(True)
            self.start_btn.Enable(True)
            self.end_btn.Enable(False)
            threading.Thread(target=self.track_men, args=(self.tracking,), daemon=True).start()
            threading.Thread(target=self.track_women, args=(self.tracking,), daemon=True).start()



class MyApp(wx.App): #Builds the frame and GUI.
    def OnInit(self):
        self.frame = MyFrame(parent=None, title="RaceTrack 1.0")
        self.frame.Show()
        return True

#Launches app.
app = MyApp(0)

app.MainLoop()
```