```
#                   TITLE BLOCK
#************************************************
#Author:    Brandon White
#Date:      08/28/2019
#Desc:      Creates a visual representation of
#           states vs time for sim data
#************************************************
import sys

from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtWidgets import QApplication, QMainWindow, QMenu,
QVBoxLayout, QSizePolicy, QMessageBox, QWidget, QPushButton
from PyQt5.QtGui import QIcon

from matplotlib.backends.backend_qt5agg import FigureCanvasQTAgg as
FigureCanvas
from matplotlib.figure import Figure
import matplotlib.pyplot as plt

import numpy, time

#Main GUI Class
class App(QMainWindow):

    def __init__(self):
        super().__init__()

        #Set form size (NOTE: graphs are [Pixels Width, Pixels
        Length] / 100)
        self.width = 1030 #40 for gaps
        self.height = 700 #50 for gapsS

        #GUI Position and Size
        self.left = 0
        self.top = 55
        self.title = 'Simulation Data Visualizer'
        self.graphs = []


    def initUI(self):
        self.setWindowTitle(self.title)
        self.setGeometry(self.left, self.top, self.width,
        self.height)

        #Create all graphs
```

```
        #create all graphs
        self.graphs.append(PlotCanvas(self, width=5, height=5,
            name_here = "position", given_data = self.data[:,0:3],
            t=self.t))
        self.graphs.append(PlotCanvas(self, width=5, height=1.6,
            name_here = "u", given_data = self.data[:, 3], t=self.t))
        self.graphs.append(PlotCanvas(self, width=5, height=1.6,
            name_here = "v", given_data = self.data[:, 4], t=self.t))
        self.graphs.append(PlotCanvas(self, width=5, height=1.6,
            name_here = "w", given_data = self.data[:, 5], t=self.t))
        self.graphs.append(PlotCanvas(self, width=3.3, height=1.8,
            name_here =  "p", given_data = self.data[:, 10],
            t=self.t))
        self.graphs.append(PlotCanvas(self, width=3.3, height=1.8,
            name_here = "q", given_data = self.data[:, 11],
            t=self.t))
        self.graphs.append(PlotCanvas(self, width=3.3, height=1.8,
            name_here = "r", given_data = self.data[:, 12],
            t=self.t))

        #Position Graphs
        self.graphs[0].move(10,0)
        self.graphs[1].move(520,0)
        self.graphs[2].move(520,170)
        self.graphs[3].move(520,340)
        self.graphs[4].move(10,510)
        self.graphs[5].move(350,510)
        self.graphs[6].move(690,510)

        for graph in self.graphs:
            graph.plot()

        self.show()

# Graphing Subclass
class PlotCanvas(FigureCanvas):

    def __init__(self, parent=None, width=5, height=4, dpi=100,
    name_here = "NO NAME GIVEN", given_data = [0], t=[0]):
        fig = Figure(figsize=(width, height), dpi=dpi)
        self.axes = fig.add_subplot(111)

        self.nombre = name_here
        self.t = t
        self.data = given_data
```

```
81
82          FigureCanvas.__init__(self, fig)
83          self.setParent(parent)
84
85          FigureCanvas.setSizePolicy(self,
86                  QSizePolicy.Expanding,
87                  QSizePolicy.Expanding)
88          FigureCanvas.updateGeometry(self)
89          self.plot()
90
91      def plot(self):
92          try:
93              if self.nombre == "position":
94                  import matplotlib.pyplot as plt
95                  from mpl_toolkits.mplot3d import Axes3D
96                  ax = self.figure.add_subplot(111, projection = '3d')
97                  ax.cla()
98                  ax.plot3D(self.data[:, 0], self.data[:, 1],
•                   self.data[:, 2])
99                  ax.set_xlabel('P_n')
100                 ax.set_ylabel('P_e')
101                 ax.set_zlabel('P_d')
102                 ax.set_title(self.nombre)
103                 ax.invert_zaxis()
104                 self.draw()
105             else:
106                 ax = self.figure.add_subplot(111)
107                 ax.plot(self.t,self.data, 'r')
108                 ax.set_title(self.nombre)
109                 self.draw()
110         except:
111             print('PLOT METHOD ERROR')
112             return
113
114 def open_GUI(t = [0], sim_data = [0,0,0,0,0,0,0,0,0,0,0,0,0,0]):
115     app = QApplication(sys.argv)
116     ex = App()
117     ex.data = sim_data
118     ex.t = t
119     ex.initUI()
120     sys.exit(app.exec_())
121
122 if __name__ == '__main__':
123     open_GUI()
124
```