Министерство образования и науки РФ

Федеральное государственное бюджетное образовательное учреждение

высшего образования «Московский политехнический университет»

**факультет информационных технологий**

**Кафедра СМАРТ-технологий**

Дисциплина: Технологии визуализации данных систем управления

Отчёт по лабораторной работе №4

«Многомерный анализ и визуализация мониторинговых данных»

Работа выполнена (ФИО):

Ф И О

Научный руководитель

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ф И О

**Цель работы**

Подготовить приложение на языке C# для сбора, статистической обработки и визуализации данных с использованием библиотеки OpenGL (через обертку SharpGL)**Задачи**

• Реализовать визуализацию сцены для отображения пространственных данных в виде куба с окружающими поверхностями для отображения графиков. Реализовать управление отображением с помощью мыши с центром вращения сцены в геометрическом центре куба. Реализовать возможность скрывать куб или отдельные поверхности для графиков.

• Реализовать загрузку набора данных, представленных как набор мониторинговых параметров из сообщений в сети CAN. Выполнить фильтрацию сообщений по заданной карте и квантование по моменту сбора данных.

• Выполнить статистический анализ записей как набора точек в многомерном пространстве, вычисляя параметры описательной статистики методом скользящих сводных параметров. Реализовать возможность задать глубину истории рассчитываемых параметров.

• Разработать систему накопления данных для отображения зависимостей трех параметров по выбору начиная с временной отсечки и в течение заданного времени накопления. Третий параметр учитывается как среднее значение на заданном интервале двух других параметров за указанный период накопления.

• Реализовать систему накопления данных для анализа зависимости трех параметров за заданный период времени, методом расчета описательной статистики (скейтерплоты) с фиксированной глубиной истории и без анализа выбросов.

• Выполнить генерацию битовых карт с графиками описательной статистики (гистограммы и скейтерплотами) для заданных параметров и реализовать их использование в качестве текстур для размещения графиков на сцене визуа- лизации.

• Реализовать систему отображения зависимости параметров как плоскости в кубе визуализации.

• Реализовать возможность отображения графиков зависимостей между двумя параметрами (описательной статистики) на поверхностях для графиков, синхронизированно с осями куба визуализации

• Реализовать систему визуализации данных (плоскости и графиков) в режиме скользящих сводных параметров.

• Реализовать отображение системы координат и ребер описывающего куба с указанием визуализируемых параметров.

• Реализовать сохранение полученного изображения в файл.

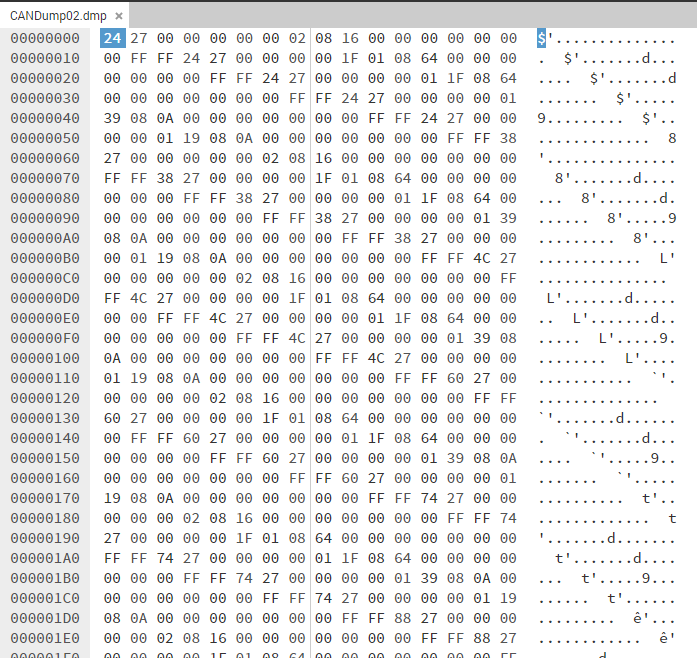
**Комментарии к выполнению**

Задача 1 Реализовать сцену с использованием средств OpenGL и созданного в предыдущих лабораторных работах кода навигации (поворота сцены) мышью.

**Ход работы**

Данные из дампа CAN-шины считываются пакетами по 19 байт в каждом. Окончанием пакета служит пара из байтов 0xFF (255)

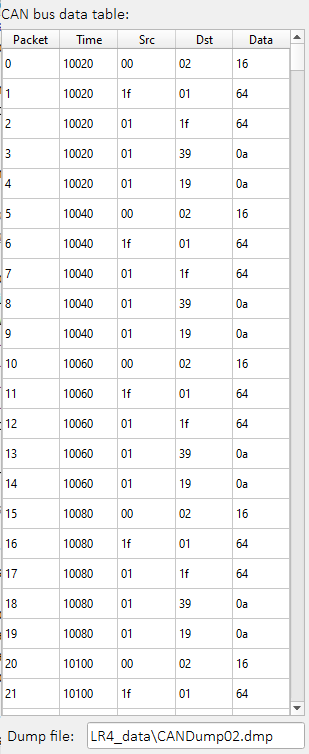
Структура файла выглядит следующим образом:



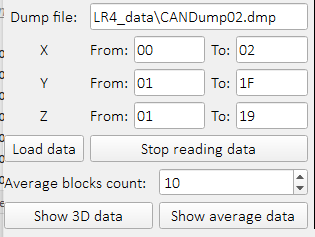
Для запуска потока чтения файла (вместо потока из файла можно использовать, например, UDP порт) на форме имеется кнопка Load data

При нажатии на эту кнопку, файл дампа открывается для чтения. А также, запускается отдельный поток, читающий отдельные байты в буфер. Как только будет обнаружена пара из символов 0xFF, буфер будет выгружен в массив данных, содержащий 2 байта времени, байт адреса отправителя, получателя и байт данных

Считанные данные отображаются в таблице на форме



В качестве координат точек (X, Y, Z) берутся данные (по 1 байту) между некоторыми двумя узлами, адрес которых можно задать в текстовых полях на форме в шестнадцеричной системе счисления

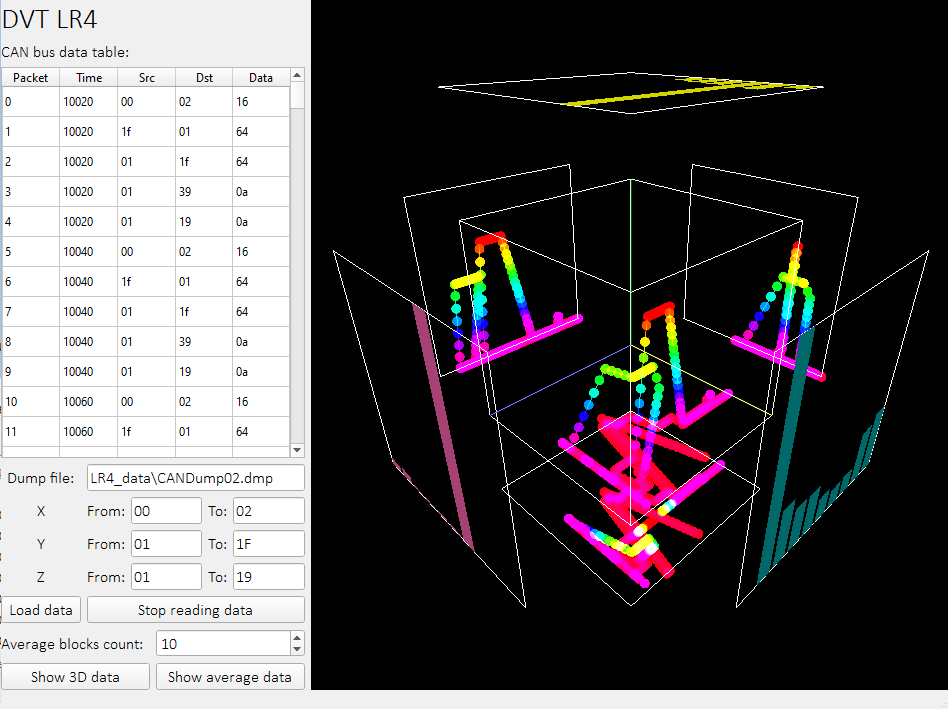


Для отображения данных на форме есть 2 кнопки:

* Show 3D data
* Show average data

При нажатии на первую кнопку, данные будут отрисованы в неизменном виде (весь массив целиком).

При нажатии второй кнопки будут отрисованы средние точки. Среднее значение вычисляется в течении указанного количества пакетов (через элемент на форме)



Перед отрисовкой весь массив данных проходит сортировку по времени. Т.к. в одно и то же время могут быть переданы несколько пакетов

Рядом с «кубом данных» имеются 3 плоскости, на которых отображаются проекции соответствующих осей (XY, YZ, ZX)

Помимо плоскостей с проекциями, с помощью OpenGL отрисовывается гистограммы плотностей для каждой проекции (плоскости)

Диаграммы вычисляются методом «скользящее окно» по 10 областей на каждую плоскость

**Вывод**

В ходе данной работы было создано приложение приложение для сбора, статистической обработки и визуализации данных с использованием библиотеки OpenGL

**Исходный код**

*"""  
This is free and unencumbered software released into the public domain.  
  
Anyone is free to copy, modify, publish, use, compile, sell, or  
distribute this software, either in source code form or as a compiled  
binary, for any purpose, commercial or non-commercial, and by any  
means.  
  
In jurisdictions that recognize copyright laws, the author or authors  
of this software dedicate any and all copyright interest in the  
software to the public domain. We make this dedication for the benefit  
of the public at large and to the detriment of our heirs and  
successors. We intend this dedication to be an overt act of  
relinquishment in perpetuity of all present and future rights to this  
software under copyright law.  
  
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,  
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF  
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.  
IN NO EVENT SHALL THE AUTHORS BE LIABLE FOR ANY CLAIM, DAMAGES OR  
OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,  
ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR  
OTHER DEALINGS IN THE SOFTWARE.  
  
For more information, please refer to <https://unlicense.org>  
"""*import os  
import sys  
import threading  
  
import matplotlib.pyplot as plt  
import numpy as np  
import pyqtgraph.opengl as gl  
from PyQt5 import uic, QtWidgets  
from PyQt5.QtWidgets import QApplication, QMainWindow, QTableWidgetItem  
  
  
class Window(QMainWindow):  
 def \_\_init\_\_(self):  
 super(Window, self).\_\_init\_\_()  
 *# Load GUI file* uic.loadUi(**'LR4.ui'**, self)  
  
 *# Settings* self.bars\_width = 0.08  
 self.bar\_z\_color = np.array(np.array([1., 0.4, 0.7, 1.]))  
 self.bar\_x\_color = np.array(np.array([1., 1., 0., 1.]))  
 self.bar\_y\_color = np.array(np.array([0., 1., 1., 1.]))  
  
 *# System variables* self.dump\_file = None  
 self.reader\_running = False  
 self.packets = []  
 self.points = []  
 self.points\_surface = gl.GLScatterPlotItem(pos=np.array([[0, 0, 0]]))  
 self.points\_line = gl.GLLinePlotItem(pos=np.array([[0, 0, 0], [0, 0, 0]]))  
 self.points\_surface\_plane\_xy = gl.GLScatterPlotItem(pos=np.array([[0, 0, 0]]))  
 self.points\_line\_plane\_xy = gl.GLLinePlotItem(pos=np.array([[0, 0, 0], [0, 0, 0]]))  
 self.points\_surface\_plane\_xz = gl.GLScatterPlotItem(pos=np.array([[0, 0, 0]]))  
 self.points\_line\_plane\_xz = gl.GLLinePlotItem(pos=np.array([[0, 0, 0], [0, 0, 0]]))  
 self.points\_surface\_plane\_yz = gl.GLScatterPlotItem(pos=np.array([[0, 0, 0]]))  
 self.points\_line\_plane\_yz = gl.GLLinePlotItem(pos=np.array([[0, 0, 0], [0, 0, 0]]))  
  
 self.bar\_graph\_x = gl.GLBarGraphItem(pos=np.array([[0, 0, 0]]), size=np.array([0, 0, 0]))  
 self.bar\_graph\_y = gl.GLBarGraphItem(pos=np.array([[0, 0, 0]]), size=np.array([0, 0, 0]))  
 self.bar\_graph\_z = gl.GLBarGraphItem(pos=np.array([[0, 0, 0]]), size=np.array([0, 0, 0]))  
  
 *# Connect GUI controls* self.btn\_load\_data.clicked.connect(self.load\_data)  
 self.btn\_stop\_reading.clicked.connect(self.stop\_reading)  
  
 self.btn\_show\_3d.clicked.connect(self.show\_3d)  
 self.btn\_show\_average.clicked.connect(self.show\_average)  
  
 *# Initialize table* self.init\_table()  
  
 *# Initialize OpenGL widget* self.init\_opengl()  
  
 *# Show GUI* self.show()  
  
 def dump\_reader(self):  
 *# Clear table and data arrays* self.points\_table.setRowCount(0)  
 self.packets = []  
  
 *# Create temp buffers* bytes\_buffer = [b'\x00'] \* 19  
 bytes\_buffer\_position = 0  
 previous\_byte = b'\x00'  
 packets\_readed = 0  
  
 *# Continue reading* while self.reader\_running:  
 incoming\_byte = self.dump\_file.read(1)  
 if incoming\_byte is None or len(incoming\_byte) == 0:  
 self.reader\_running = False  
 else:  
 bytes\_buffer[bytes\_buffer\_position] = incoming\_byte  
 if bytes\_buffer[bytes\_buffer\_position] == b'\xff' and previous\_byte == b'\xff':  
 bytes\_buffer\_position = 0  
  
 time = int.from\_bytes(b''.join([bytes\_buffer[1], bytes\_buffer[0]]), byteorder=**'big'**, signed=False)  
 source = bytes\_buffer[6]  
 destination = bytes\_buffer[7]  
 data = bytes\_buffer[9]  
  
 self.packets.append([time, source, destination, data])  
  
 self.points\_table.insertRow(packets\_readed)  
 self.points\_table.setItem(packets\_readed, 0, QTableWidgetItem(str(packets\_readed)))  
 self.points\_table.setItem(packets\_readed, 1, QTableWidgetItem(str(time)))  
 self.points\_table.setItem(packets\_readed, 2, QTableWidgetItem(str(source.hex())))  
 self.points\_table.setItem(packets\_readed, 3, QTableWidgetItem(str(destination.hex())))  
 self.points\_table.setItem(packets\_readed, 4, QTableWidgetItem(str(data.hex())))  
  
 packets\_readed += 1  
 else:  
 previous\_byte = bytes\_buffer[bytes\_buffer\_position]  
 bytes\_buffer\_position += 1  
 if bytes\_buffer\_position >= 19:  
 bytes\_buffer\_position = 0  
  
 self.dump\_file.close()  
 print(**'File reading stopped. Readed'**, packets\_readed, **'packets'**)  
  
 def init\_table(self):  
 *"""  
 Initializes table of points  
 :return:  
 """* self.points\_table.setColumnCount(5)  
 self.points\_table.verticalHeader().setVisible(False)  
 self.points\_table.setEditTriggers(QtWidgets.QAbstractItemView.NoEditTriggers)  
 self.points\_table.setHorizontalHeaderItem(0, QtWidgets.QTableWidgetItem(**'Packet'**))  
 self.points\_table.setHorizontalHeaderItem(1, QtWidgets.QTableWidgetItem(**'Time'**))  
 self.points\_table.setHorizontalHeaderItem(2, QtWidgets.QTableWidgetItem(**'Src'**))  
 self.points\_table.setHorizontalHeaderItem(3, QtWidgets.QTableWidgetItem(**'Dst'**))  
 self.points\_table.setHorizontalHeaderItem(4, QtWidgets.QTableWidgetItem(**'Data'**))  
 header = self.points\_table.horizontalHeader()  
 header.setSectionResizeMode(0, QtWidgets.QHeaderView.Stretch)  
 header.setSectionResizeMode(1, QtWidgets.QHeaderView.Stretch)  
 header.setSectionResizeMode(2, QtWidgets.QHeaderView.Stretch)  
 header.setSectionResizeMode(3, QtWidgets.QHeaderView.Stretch)  
 header.setSectionResizeMode(4, QtWidgets.QHeaderView.Stretch)  
  
 def init\_opengl(self):  
 *"""  
 Initializes OpenGL Widget  
 :return:  
 """  
  
 # Cube bottom square* cube\_bottom\_square = gl.GLLinePlotItem(  
 pos=np.array([[0, 0, 0], [1, 0, 0], [1, 1, 0], [0, 1, 0], [0, 0, 0]]),  
 color=[1, 1, 1, 1])  
 self.openGLWidget.addItem(cube\_bottom\_square)  
  
 *# Cube top square* cube\_top\_square = gl.GLLinePlotItem(  
 pos=np.array([[0, 0, 1], [1, 0, 1], [1, 1, 1], [0, 1, 1], [0, 0, 1]]),  
 color=[1, 1, 1, 1])  
 self.openGLWidget.addItem(cube\_top\_square)  
  
 *# Cube sides* cube\_line\_bl = gl.GLLinePlotItem(  
 pos=np.array([[0, 0, 0], [0, 0, 1]]),  
 color=[1, 1, 1, 1])  
 self.openGLWidget.addItem(cube\_line\_bl)  
 cube\_line\_br = gl.GLLinePlotItem(  
 pos=np.array([[0, 1, 0], [0, 1, 1]]),  
 color=[1, 1, 1, 1])  
 self.openGLWidget.addItem(cube\_line\_br)  
 cube\_line\_tr = gl.GLLinePlotItem(  
 pos=np.array([[1, 1, 0], [1, 1, 1]]),  
 color=[1, 1, 1, 1])  
 self.openGLWidget.addItem(cube\_line\_tr)  
 cube\_line\_tl = gl.GLLinePlotItem(  
 pos=np.array([[1, 0, 0], [1, 0, 1]]),  
 color=[1, 1, 1, 1])  
 self.openGLWidget.addItem(cube\_line\_tl)  
  
 *# Planes* plane\_xzp = gl.GLLinePlotItem(  
 pos=np.array([[0, -0.5, 0], [1, -0.5, 0], [1, -0.5, 1], [0, -0.5, 1], [0, -0.5, 0]]),  
 color=[1, 1, 1, 1])  
 self.openGLWidget.addItem(plane\_xzp)  
 plane\_xzm = gl.GLLinePlotItem(  
 pos=np.array([[0, 1.5, 0], [1, 1.5, 0], [1, 1.5, 1], [0, 1.5, 1], [0, 1.5, 0]]),  
 color=[1, 1, 1, 1])  
 self.openGLWidget.addItem(plane\_xzm)  
 plane\_xyp = gl.GLLinePlotItem(  
 pos=np.array([[0, 0, 1.5], [1, 0, 1.5], [1, 1, 1.5], [0, 1, 1.5], [0, 0, 1.5]]),  
 color=[1, 1, 1, 1])  
 self.openGLWidget.addItem(plane\_xyp)  
 plane\_xym = gl.GLLinePlotItem(  
 pos=np.array([[0, 0, -0.5], [1, 0, -0.5], [1, 1, -0.5], [0, 1, -0.5], [0, 0, -0.5]]),  
 color=[1, 1, 1, 1])  
 self.openGLWidget.addItem(plane\_xym)  
 plane\_yzp = gl.GLLinePlotItem(  
 pos=np.array([[1.5, 0, 0], [1.5, 1, 0], [1.5, 1, 1], [1.5, 0, 1], [1.5, 0, 0]]),  
 color=[1, 1, 1, 1])  
 self.openGLWidget.addItem(plane\_yzp)  
 plane\_yzm = gl.GLLinePlotItem(  
 pos=np.array([[-0.5, 0, 0], [-0.5, 1, 0], [-0.5, 1, 1], [-0.5, 0, 1], [-0.5, 0, 0]]),  
 color=[1, 1, 1, 1])  
 self.openGLWidget.addItem(plane\_yzm)  
  
 *# Add data elements* self.openGLWidget.addItem(gl.GLAxisItem())  
 self.openGLWidget.addItem(self.points\_surface)  
 self.openGLWidget.addItem(self.points\_line)  
 self.openGLWidget.addItem(self.points\_surface\_plane\_xy)  
 self.openGLWidget.addItem(self.points\_line\_plane\_xy)  
 self.openGLWidget.addItem(self.points\_surface\_plane\_xz)  
 self.openGLWidget.addItem(self.points\_line\_plane\_xz)  
 self.openGLWidget.addItem(self.points\_surface\_plane\_yz)  
 self.openGLWidget.addItem(self.points\_line\_plane\_yz)  
 self.openGLWidget.addItem(self.bar\_graph\_x)  
 self.openGLWidget.addItem(self.bar\_graph\_y)  
 self.openGLWidget.addItem(self.bar\_graph\_z)  
  
 def show\_3d(self, average=False):  
 if average:  
 blocks = self.average\_blocks.value()  
 else:  
 blocks = 0  
 if len(self.packets) > 1:  
 blocks\_counter = 0  
 x = 0  
 y = 0  
 z = 0  
 x\_count = 0  
 y\_count = 0  
 z\_count = 0  
  
 self.points = []  
 *# Sort packets by time* self.packets = sorted(self.packets, key=lambda x: (x[0]))  
 time = -1  
 for i in range(len(self.packets)):  
 if self.packets[i][0] != time:  
 *# New block* time = self.packets[i][0]  
 k = i  
 while k < len(self.packets) and self.packets[k][0] == time:  
 source = self.packets[k][1]  
 destination = self.packets[k][2]  
 data = self.packets[k][3]  
 *# X* if source == bytes.fromhex(self.line\_x\_from.text()) and \  
 destination == bytes.fromhex(self.line\_x\_to.text()):  
 x += float(int(data.hex(), 16))  
 x\_count += 1  
  
 *# Y* if source == bytes.fromhex(self.line\_y\_from.text()) and \  
 destination == bytes.fromhex(self.line\_y\_to.text()):  
 y += float(int(data.hex(), 16))  
 y\_count += 1  
  
 *# Z* if source == bytes.fromhex(self.line\_z\_from.text()) and \  
 destination == bytes.fromhex(self.line\_z\_to.text()):  
 z += float(int(data.hex(), 16))  
 z\_count += 1  
  
 k += 1  
  
 blocks\_counter += 1  
 if blocks\_counter > blocks:  
 if x\_count > 0:  
 x /= x\_count  
 if y\_count > 0:  
 y /= y\_count  
 if z\_count > 0:  
 z /= z\_count  
 self.points.append([x, y, z])  
 blocks\_counter = 0  
 x = 0  
 y = 0  
 z = 0  
 x\_count = 0  
 y\_count = 0  
 z\_count = 0  
  
 self.points = np.array(self.points)  
 self.points /= np.max(self.points)  
 self.draw\_points()  
  
 def show\_average(self):  
 self.show\_3d(True)  
  
 def show\_on\_table(self):  
 *"""  
 Shows points in table  
 :return:  
 """* self.points\_table.setRowCount(0)  
 for point in self.points:  
 row\_position = self.points\_table.rowCount()  
 self.points\_table.insertRow(row\_position)  
 self.points\_table.setItem(row\_position, 0, QTableWidgetItem(str(point[0])))  
 self.points\_table.setItem(row\_position, 1, QTableWidgetItem(str(point[1])))  
 self.points\_table.setItem(row\_position, 2, QTableWidgetItem(str(point[2])))  
  
 def update\_bars(self):  
 *"""  
 Updates and draws 3D - bar charts  
 """* bar\_pos, bar\_size = self.count\_blocks([item[1] for item in self.points], 0)  
 self.openGLWidget.removeItem(self.bar\_graph\_x)  
 self.bar\_graph\_x = gl.GLBarGraphItem(pos=np.array(bar\_pos), size=np.array(bar\_size))  
 self.bar\_graph\_x.setColor(self.bar\_x\_color)  
 self.openGLWidget.addItem(self.bar\_graph\_x)  
  
 bar\_pos, bar\_size = self.count\_blocks([item[0] for item in self.points], 1)  
 self.openGLWidget.removeItem(self.bar\_graph\_y)  
 self.bar\_graph\_y = gl.GLBarGraphItem(pos=np.array(bar\_pos), size=np.array(bar\_size))  
 self.bar\_graph\_y.setColor(self.bar\_y\_color)  
 self.openGLWidget.addItem(self.bar\_graph\_y)  
  
 bar\_pos, bar\_size = self.count\_blocks([item[2] for item in self.points], 2)  
 self.openGLWidget.removeItem(self.bar\_graph\_z)  
 self.bar\_graph\_z = gl.GLBarGraphItem(pos=np.array(bar\_pos[::-1]), size=np.array(bar\_size))  
 self.bar\_graph\_z.setColor(self.bar\_z\_color)  
 self.openGLWidget.addItem(self.bar\_graph\_z)  
  
 def count\_blocks(self, data, orientation):  
 *"""  
 Counts how many points in each block and returns bars data  
 """* max\_block\_value = 0  
 bar\_pos = []  
 bar\_size = []  
 for x in range(0, 10):  
 points\_in\_block = 0  
 for i in range(len(data)):  
 if x / 10 <= data[i] < (x / 10 + 0.1):  
 points\_in\_block += 1  
 if points\_in\_block > max\_block\_value:  
 max\_block\_value = points\_in\_block  
  
 if points\_in\_block > 0:  
 if orientation == 0:  
 pos = [0, x / 10, 1.5]  
 size = [points\_in\_block, self.bars\_width, 0]  
 elif orientation == 1:  
 pos = [x / 10, 1.5, 0]  
 size = [self.bars\_width, 0, points\_in\_block]  
 else:  
 pos = [1.5, x / 10, 0]  
 size = [0, self.bars\_width, points\_in\_block]  
 bar\_pos.append(pos)  
 bar\_size.append(size)  
  
 bar\_size\_normalized = []  
 for bar in bar\_size:  
 if orientation == 0:  
 bar[0] /= max\_block\_value  
 elif orientation == 1:  
 bar[2] /= max\_block\_value  
 elif orientation == 2:  
 bar[2] /= max\_block\_value  
 bar\_size\_normalized.append(bar)  
 return bar\_pos, bar\_size\_normalized  
  
 def draw\_points(self):  
 *"""  
 Draws 3D points with OpenGL  
 :return:  
 """* if self.points is not None and len(self.points) > 0:  
 points\_plane\_xy = np.array([[item[0], item[1], -0.5] for item in self.points])  
 points\_plane\_xz = np.array([[item[0], -0.5, item[2]] for item in self.points])  
 points\_plane\_yz = np.array([[-0.5, item[1], item[2]] for item in self.points])  
 *# Create color map* z = np.array(np.array([item[2] for item in self.points]))  
 cmap = plt.get\_cmap(**'hsv'**)  
 min\_z = np.min(z)  
 max\_z = np.max(z)  
 if max\_z != min\_z:  
 rgba\_img = cmap(1.0 - (z - min\_z) / (max\_z - min\_z))  
 else:  
 rgba\_img = cmap(1.0 - (z - min\_z))  
  
 *# Draw points* self.points\_surface.setData(pos=self.points, color=rgba\_img)  
 self.points\_line.setData(pos=self.points, color=rgba\_img)  
 self.points\_surface\_plane\_xy.setData(pos=points\_plane\_xy, color=rgba\_img)  
 self.points\_line\_plane\_xy.setData(pos=points\_plane\_xy, color=rgba\_img)  
 self.points\_surface\_plane\_xz.setData(pos=points\_plane\_xz, color=rgba\_img)  
 self.points\_line\_plane\_xz.setData(pos=points\_plane\_xz, color=rgba\_img)  
 self.points\_surface\_plane\_yz.setData(pos=points\_plane\_yz, color=rgba\_img)  
 self.points\_line\_plane\_yz.setData(pos=points\_plane\_yz, color=rgba\_img)  
  
 self.update\_bars()  
  
 def load\_data(self):  
 *"""  
 Loads dump file  
 :return:  
 """* if not self.reader\_running:  
 if os.path.exists(self.data\_file.text()):  
 print(**'Loading data...'**)  
 self.dump\_file = open(self.data\_file.text(), **'rb'**)  
 self.reader\_running = True  
 thread = threading.Thread(target=self.dump\_reader)  
 thread.start()  
 else:  
 print(**'File'**, self.data\_file.text(), **'doesn**\'**t exist!'**)  
  
 def stop\_reading(self):  
 *"""  
 Stops reading data from dump file  
 :return:  
 """* self.reader\_running = False  
  
  
if \_\_name\_\_ == **'\_\_main\_\_'**:  
 app = QApplication(sys.argv)  
 app.setStyle(**'fusion'**)  
 win = Window()  
 sys.exit(app.exec\_())

*<?*xml version="1.0" encoding="UTF-8"*?>  
<!--  
This is free and unencumbered software released into the public domain.  
  
Anyone is free to copy, modify, publish, use, compile, sell, or  
distribute this software, either in source code form or as a compiled  
binary, for any purpose, commercial or non-commercial, and by any  
means.  
  
In jurisdictions that recognize copyright laws, the author or authors  
of this software dedicate any and all copyright interest in the  
software to the public domain. We make this dedication for the benefit  
of the public at large and to the detriment of our heirs and  
successors. We intend this dedication to be an overt act of  
relinquishment in perpetuity of all present and future rights to this  
software under copyright law.  
  
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,  
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF  
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.  
IN NO EVENT SHALL THE AUTHORS BE LIABLE FOR ANY CLAIM, DAMAGES OR  
OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,  
ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR  
OTHER DEALINGS IN THE SOFTWARE.  
  
For more information, please refer to <https://unlicense.org>  
 -->*<ui version="4.0">  
 <class>MainWindow</class>  
 <widget class="QMainWindow" name="MainWindow">  
 <property name="geometry">  
 <rect>  
 <x>0</x>  
 <y>0</y>  
 <width>950</width>  
 <height>712</height>  
 </rect>  
 </property>  
 <property name="font">  
 <font>  
 <family>Calibri Light</family>  
 <pointsize>12</pointsize>  
 </font>  
 </property>  
 <property name="windowTitle">  
 <string>DVT LR4</string>  
 </property>  
 <widget class="QWidget" name="centralwidget">  
 <layout class="QGridLayout" name="gridLayout">  
 <item row="0" column="0">  
 <layout class="QVBoxLayout" name="verticalLayout\_3">  
 <item>  
 <widget class="QLabel" name="label">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Fixed" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="font">  
 <font>  
 <pointsize>22</pointsize>  
 </font>  
 </property>  
 <property name="text">  
 <string>DVT LR4</string>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QLabel" name="label\_4">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Fixed" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>CAN bus data table:</string>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QTableWidget" name="points\_table">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Minimum" vsizetype="Expanding">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="sortingEnabled">  
 <bool>true</bool>  
 </property>  
 <attribute name="horizontalHeaderShowSortIndicator" stdset="0">  
 <bool>true</bool>  
 </attribute>  
 <attribute name="verticalHeaderShowSortIndicator" stdset="0">  
 <bool>false</bool>  
 </attribute>  
 </widget>  
 </item>  
 <item>  
 <layout class="QGridLayout" name="gridLayout\_4">  
 <item row="3" column="1">  
 <layout class="QHBoxLayout" name="horizontalLayout\_2">  
 <item>  
 <widget class="QLabel" name="label\_10">  
 <property name="text">  
 <string>From:</string>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QLineEdit" name="line\_y\_from">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Preferred" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>01</string>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QLabel" name="label\_11">  
 <property name="text">  
 <string>To:</string>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QLineEdit" name="line\_y\_to">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Preferred" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>1F</string>  
 </property>  
 </widget>  
 </item>  
 </layout>  
 </item>  
 <item row="5" column="0">  
 <widget class="QPushButton" name="btn\_load\_data">  
 <property name="text">  
 <string>Load data</string>  
 </property>  
 </widget>  
 </item>  
 <item row="5" column="1">  
 <widget class="QPushButton" name="btn\_stop\_reading">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Preferred" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>Stop reading data</string>  
 </property>  
 </widget>  
 </item>  
 <item row="1" column="0">  
 <widget class="QLabel" name="label\_19">  
 <property name="text">  
 <string>Dump file:</string>  
 </property>  
 <property name="alignment">  
 <set>Qt::AlignCenter</set>  
 </property>  
 </widget>  
 </item>  
 <item row="1" column="1">  
 <widget class="QLineEdit" name="data\_file">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Preferred" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>LR4\_data\CANDump02.dmp</string>  
 </property>  
 </widget>  
 </item>  
 <item row="2" column="0">  
 <widget class="QLabel" name="label\_5">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Preferred" vsizetype="Preferred">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>X</string>  
 </property>  
 <property name="alignment">  
 <set>Qt::AlignCenter</set>  
 </property>  
 </widget>  
 </item>  
 <item row="3" column="0">  
 <widget class="QLabel" name="label\_6">  
 <property name="text">  
 <string>Y</string>  
 </property>  
 <property name="alignment">  
 <set>Qt::AlignCenter</set>  
 </property>  
 </widget>  
 </item>  
 <item row="4" column="0">  
 <widget class="QLabel" name="label\_7">  
 <property name="text">  
 <string>Z</string>  
 </property>  
 <property name="alignment">  
 <set>Qt::AlignCenter</set>  
 </property>  
 </widget>  
 </item>  
 <item row="2" column="1">  
 <layout class="QHBoxLayout" name="horizontalLayout">  
 <item>  
 <widget class="QLabel" name="label\_8">  
 <property name="text">  
 <string>From:</string>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QLineEdit" name="line\_x\_from">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Preferred" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>00</string>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QLabel" name="label\_9">  
 <property name="text">  
 <string>To:</string>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QLineEdit" name="line\_x\_to">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Preferred" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>02</string>  
 </property>  
 </widget>  
 </item>  
 </layout>  
 </item>  
 <item row="4" column="1">  
 <layout class="QHBoxLayout" name="horizontalLayout\_3">  
 <item>  
 <widget class="QLabel" name="label\_12">  
 <property name="text">  
 <string>From:</string>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QLineEdit" name="line\_z\_from">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Preferred" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>01</string>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QLabel" name="label\_13">  
 <property name="text">  
 <string>To:</string>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QLineEdit" name="line\_z\_to">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Preferred" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>19</string>  
 </property>  
 </widget>  
 </item>  
 </layout>  
 </item>  
 </layout>  
 </item>  
 <item>  
 <layout class="QGridLayout" name="gridLayout\_2">  
 <item row="0" column="0">  
 <widget class="QLabel" name="label\_3">  
 <property name="text">  
 <string>Average blocks count:</string>  
 </property>  
 </widget>  
 </item>  
 <item row="0" column="1">  
 <widget class="QSpinBox" name="average\_blocks">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Preferred" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="value">  
 <number>10</number>  
 </property>  
 </widget>  
 </item>  
 <item row="1" column="0">  
 <widget class="QPushButton" name="btn\_show\_3d">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Preferred" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>Show 3D data</string>  
 </property>  
 </widget>  
 </item>  
 <item row="1" column="1">  
 <widget class="QPushButton" name="btn\_show\_average">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Preferred" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>Show average data</string>  
 </property>  
 </widget>  
 </item>  
 </layout>  
 </item>  
 </layout>  
 </item>  
 <item row="0" column="1">  
 <layout class="QVBoxLayout" name="verticalLayout\_4">  
 <item>  
 <widget class="GLViewWidget" name="openGLWidget">  
 <property name="minimumSize">  
 <size>  
 <width>640</width>  
 <height>480</height>  
 </size>  
 </property>  
 </widget>  
 </item>  
 </layout>  
 </item>  
 </layout>  
 </widget>  
 <widget class="QMenuBar" name="menubar">  
 <property name="geometry">  
 <rect>  
 <x>0</x>  
 <y>0</y>  
 <width>950</width>  
 <height>21</height>  
 </rect>  
 </property>  
 </widget>  
 <widget class="QStatusBar" name="statusbar"/>  
 </widget>  
 <customwidgets>  
 <customwidget>  
 <class>GLViewWidget</class>  
 <extends>QOpenGLWidget</extends>  
 <header>pyqtgraph.opengl</header>  
 </customwidget>  
 </customwidgets>  
 <resources/>  
 <connections/>  
</ui>