Actividad 11 (Fuerza Bruta)

Jose Eduardo Silva Canizales

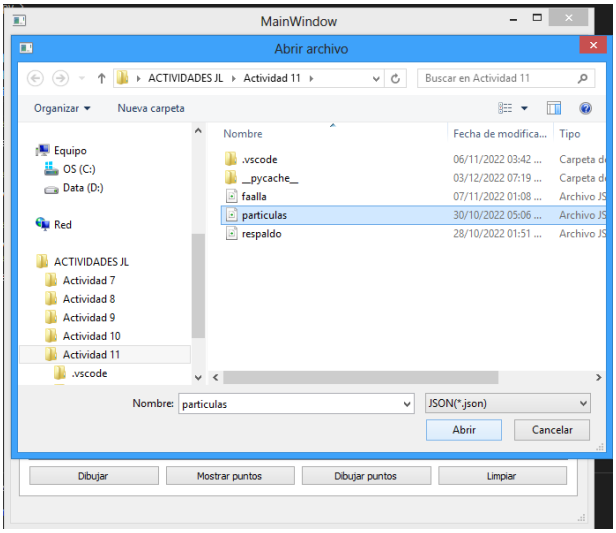
Seminario de solución de problemas de algoritmia

Lineamientos de evaluación

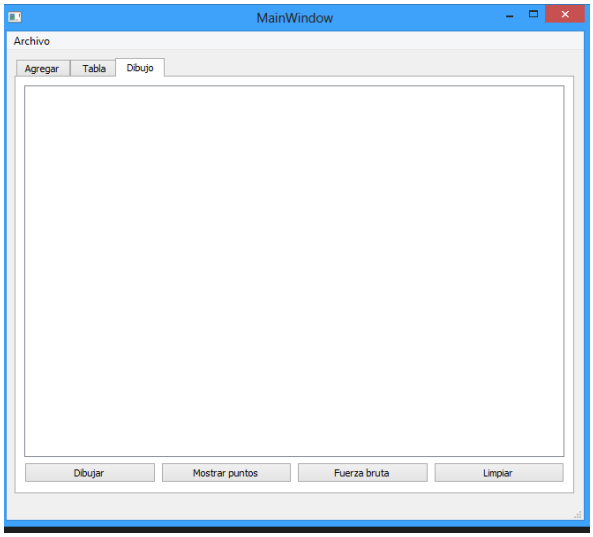
- El reporte está en formato Google Docs o PDF.
- El reporte sigue las pautas del Formato de Actividades.
- El reporte tiene desarrollada todas las pautas del Formato de Actividades.
- Se muestra captura de pantalla de los puntos de las partículas en el QScene.
- Se muestra captura de pantalla del resultado del algoritmo de fuerza bruta en el QScene.

Desarrollo

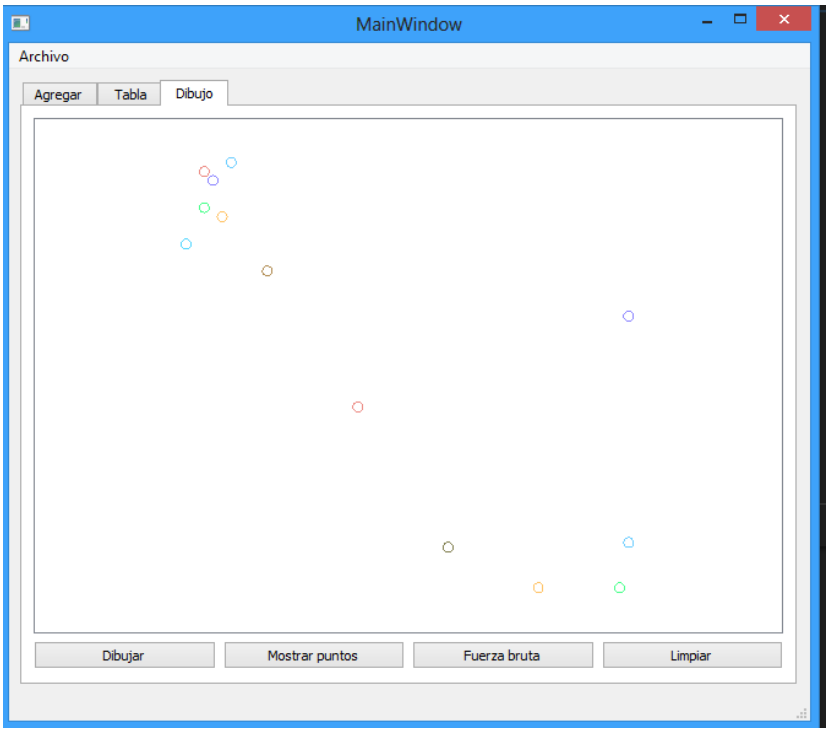
Captura de pantalla de archivo seleccionado.



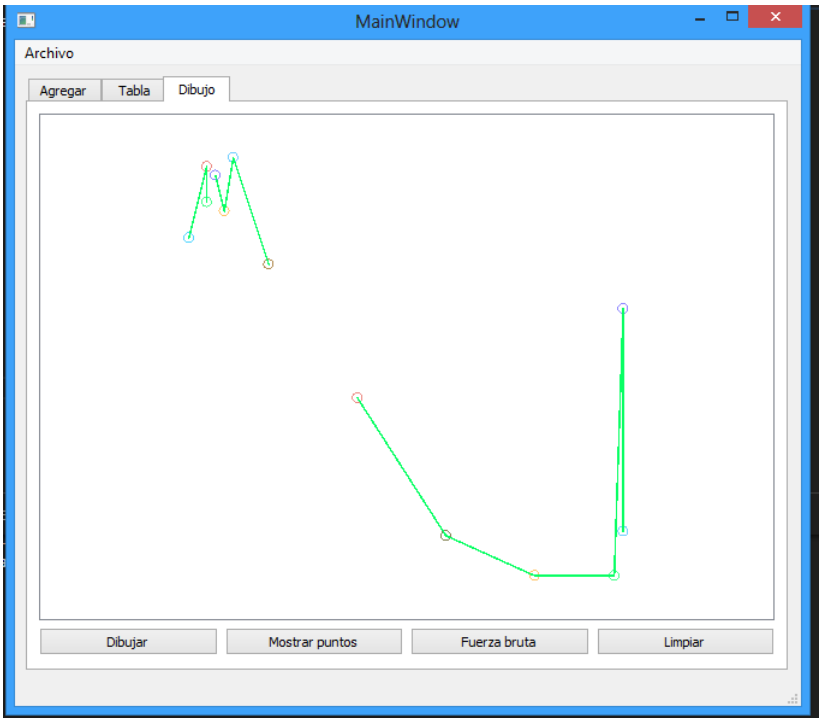
Captura de pantalla del Qscene vacío.



Captura de pantalla de los puntos de las partículas en el QScene.



Captura de pantalla del resultado del algoritmo de fuerza bruta en el QScene.



Conclusiones

Durante esta actividad se implementó algoritmos para que se visualizara únicamente los puntos de las partículas por otra parte aprendí e implemente el algoritmo de fuerza bruta para unir los puntos de las partículas con el más cercano y se visualizara en el QScene.

Por otra parte lo que se me complico durante esta actividad fue el poder integrar el algoritmo de fuerza bruta con las actividades previamente hechas ya que no podía hacer que se visualizara las partículas previamente ingresadas, tuve que consultar mas videos y paginas para la implementación del algoritmo de manera correcta.

Referencias

Primera referencia

Url: https://www.youtube.com/watch?v=hPARP71VEH0

Título: Actividad 11 (Fuerza Bruta)

Autor: MICHEL DAVALOS BOITES

Segunda referencia

Url: https://www.youtube.com/watch?v=NUsjpfKaD_U

Título: Clase Fuerza Bruta (18-oct-21)

Autor: MICHEL DAVALOS BOITES

Código

main.py

```
from PySide2.QtWidgets import  QApplication
from mainwindow import MainWindow
import sys

# Aplicacion de Qt
app = QApplication()

# Se crea window
window = MainWindow()

#se hace visible window
window.show()
# Qt loop
sys.exit(app.exec_())
```

mainwindow.py

```
from wsgiref import headers
from PySide2.QtWidgets import
QMainWindow,QFileDialog,QMessageBox,QTableWidgetItem,QGraphicsScene
from PySide2.QtCore import Slot
from PySide2.QtGui import QPen,QColor,QTransform
from ui_mainwindow import Ui_MainWindow
from particula import Particula
from administrador import Adminisrador
from random import randint
from pprint import pprint
from algoritmos import distancia_euclidiana
from cmath import sqrt
import math



class MainWindow(QMainWindow):
    def __init__(self):
        super(MainWindow,self).__init__()

        self.administrador=Adminisrador()
        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)

        self.puntos = []

        self.ui.agregar_final_pushButton.clicked.connect(
            self.click_agregar_final)
        self.ui.agregar_inicio_pushButton.clicked.connect(
            self.click_agregar_inicio)
        self.ui.mostar_pushButton.clicked.connect(self.click_mostrar)

        self.ui.actionAbrir.triggered.connect(self.action_abrir_archivo)
```

```
        self.ui.actionGuardar.triggered.connect(self.action_guardar_archivo)

        self.ui.mostrar_tabla_pushButton.clicked.connect(self.mostrar_tabla)
        self.ui.buscar_pushButton.clicked.connect(self.buscar_id)

        self.ui.dibujar_pushButton.clicked.connect(self.dibujar)
        self.ui.limpiar_pushButton.clicked.connect(self.limpiar)

        self.ui.plainText_id_pushButton.clicked.connect(self.ordenar_id)
        self.ui.plainText_distancia_pushButton.clicked.connect(
            self.ordenar_distancia)
        self.ui.plainText_velocidad_pushButton.clicked.connect(
            self.ordenar_velocidad)

        self.ui.table_id_pushButton.clicked.connect(self.ordenar_id)
        self.ui.table_distancia_pushButton.clicked.connect(
            self.ordenar_distancia)

        self.ui.table_velocidad_pushButton.clicked.connect(
            self.ordenar_velocidad)

        self.ui.mostrar_puntos_pushButton.clicked.connect(self.mostrar_punto
s)
        self.ui.dibujar_puntos_pushButton.clicked.connect(self.fuerza_bruta)


        self.scene=QGraphicsScene()
        self.ui.graphicsView.setScene(self.scene)



    def wheelEvent(self,event):
            if event.delta() > 0:
                self.ui.graphicsView.scale(1.2,1.2)
            else:
                self.ui.graphicsView.scale(0.8,0.8)


    @Slot()
    def fuerza_bruta(self):
        pen=QPen()
        pen.setWidth(2)

        for particula in self.administrador:
            color = QColor(int(particula.red), int(
                particula.green), int(particula.blue))
            pen.setColor(color)
            resultado=self.administrador.punto()


            for punto1,punto2 in resultado:

                x1=punto1[0]
```

```
                y1=punto1[1]
                x2=punto2[0]
                y2=punto2[1]
                self.scene.addLine(x1+6,y1+6,x2+6,y2+6,pen)



    @Slot()
    def mostrar_puntos(self):
         self.scene.clear()
         if len(self.administrador) > 0:
            pen=QPen()
            self.puntos = self.get_puntos()
            pprint(self.puntos)
            for particula in self.administrador:
                color=QColor(int(particula.red),int(particula.green),int(par
ticula.blue))
                pen.setColor(color)

                self.scene.addEllipse(particula.origen_x,
                                      particula.origen_y, 11, 11,pen)
                self.scene.addEllipse(particula.destino_x,
                                      particula.destino_y, 11, 11,pen)



    @Slot()
    def get_puntos(self):
        for particula in self.administrador:
             x = (particula.origen_x,particula.destino_x)
             y = (particula.origen_y, particula.destino_y)

             self.puntos.append((x,y))
        return self.puntos






    def ordenar_id(self):  # Manda a llamar un metodo que se encuentra en la
clase administradora
        self.administrador.ordenar_ID()
        self.click_mostrar()
        self.mostrar_tabla()

    @Slot()
    # Manda a llamar un metodo que se encuentra en la clase administradora
    def ordenar_distancia(self):
        self.administrador.ordenar_DISTANCIA()
        self.click_mostrar()
        self.mostrar_tabla()
```

```
    @Slot()
    # Manda a llamar un metodo que se encuentra en la clase administradora
    def ordenar_velocidad(self):
        self.administrador.ordenar_VELOCIDAD()
        self.click_mostrar()
        self.mostrar_tabla()


    @Slot()
    def dibujar(self):
       self.scene.clear()
       if len(self.administrador) > 0:
        pen=QPen()

        for particula in self.administrador:
            color=QColor(int(particula.red),int(particula.green),int(particu
la.blue))
            pen.setColor(color)

            dimension=4
            pen.setWidth(dimension)

            self.scene.addLine(particula.origen_x+3,particula.origen_y+3,par
ticula.destino_x,particula.destino_y,pen)
            self.scene.addEllipse(particula.origen_x,
                                  particula.origen_y, dimension,
dimension,pen)
            self.scene.addEllipse(particula.destino_x,
                                  particula.destino_y, dimension, dimension,
pen)


    @Slot()
    def limpiar(self):
        self.scene.clear()


    @Slot()
    def buscar_id(self):
        #value
        id = self.ui.buscar_lineEdit.text()
        encontrado=False
        for particula in self.administrador:
            if id== particula.id:
                self.ui.tabla.clear()
                self.ui.tabla.setRowCount(1)

                id_widget = QTableWidgetItem(particula.id)
                origen_x_widget = QTableWidgetItem(str(particula.origen_x))
                origen_y_widget = QTableWidgetItem(str(particula.origen_y))
                destino_x_widget =
QTableWidgetItem(str(particula.destino_x))
```

```
                destino_y_widget =
QTableWidgetItem(str(particula.destino_y))
                velocidad_widget =
QTableWidgetItem(str(particula.velocidad))
                red_widget = QTableWidgetItem(str(particula.red))
                green_widget = QTableWidgetItem(str(particula.green))
                blue_widget = QTableWidgetItem(str(particula.blue))
                distancia_widget =
QTableWidgetItem(str(particula.distancia))

                self.ui.tabla.setItem(0, 0, id_widget)
                self.ui.tabla.setItem(0, 1, origen_x_widget)
                self.ui.tabla.setItem(0, 2, origen_y_widget)
                self.ui.tabla.setItem(0, 3, destino_x_widget)
                self.ui.tabla.setItem(0, 4, destino_y_widget)
                self.ui.tabla.setItem(0, 5, velocidad_widget)
                self.ui.tabla.setItem(0, 6, red_widget)
                self.ui.tabla.setItem(0, 7, green_widget)
                self.ui.tabla.setItem(0, 8, blue_widget)
                self.ui.tabla.setItem(0, 9, distancia_widget)

                encontrado=True

                return
        if  not encontrado:
            QMessageBox.warning(
                self,
                "Atencion",
                f'La particula con el id "{id}" no fue encontrado'
            )




    @Slot()
    def mostrar_tabla(self):
        self.ui.tabla.setColumnCount(10)
        headers =
["Id","Origen_x","Origen_y","Destino_x","Destino_y","Velocidad","Red","Green
","Blue","Distancia"]
        self.ui.tabla.setHorizontalHeaderLabels(headers)

        self.ui.tabla.setRowCount(len(self.administrador))



        row=0
        for particula in self.administrador:

            id_widget = QTableWidgetItem(particula.id)
            origen_x_widget = QTableWidgetItem(str(particula.origen_x))
            origen_y_widget = QTableWidgetItem(str(particula.origen_y))
            destino_x_widget = QTableWidgetItem(str(particula.destino_x))
```

```
            destino_y_widget = QTableWidgetItem(str(particula.destino_y))
            velocidad_widget = QTableWidgetItem(str(particula.velocidad))
            red_widget = QTableWidgetItem(str(particula.red))
            green_widget = QTableWidgetItem(str(particula.green))
            blue_widget = QTableWidgetItem(str(particula.blue))
            distancia_widget = QTableWidgetItem(str(particula.distancia))

            self.ui.tabla.setItem(row, 0, id_widget)
            self.ui.tabla.setItem(row, 1, origen_x_widget)
            self.ui.tabla.setItem(row, 2, origen_y_widget)
            self.ui.tabla.setItem(row, 3, destino_x_widget)
            self.ui.tabla.setItem(row, 4, destino_y_widget)
            self.ui.tabla.setItem(row, 5, velocidad_widget)
            self.ui.tabla.setItem(row, 6, red_widget)
            self.ui.tabla.setItem(row, 7, green_widget)
            self.ui.tabla.setItem(row, 8, blue_widget)
            self.ui.tabla.setItem(row, 9, distancia_widget)

            row+=1




@Slot()
def action_abrir_archivo(self):
    ubicacion=QFileDialog.getOpenFileName(
        self,
        'Abrir archivo',
        '.',
        'JSON(*.json)'
    )[0]
    print(ubicacion)
    if self.administrador.abrir(ubicacion):
        QMessageBox.information(
            self,
            "Exito",
            "se cargo el archivo" + ubicacion
        )
    else:
        QMessageBox.critical(
            self,
            "Error",
            "Error al abrir el archivo" + ubicacion
        )



@Slot()
def action_guardar_archivo(self):
    #print('guardar_archivo')
    ubicacion=QFileDialog.getSaveFileName(
        self,
        'Guardar',
```

```
            '.',
            'JSON(*.json)'
        )[0]
        print(ubicacion)
        if self.administrador.guardar(ubicacion):
            QMessageBox.information(
                self,
                "Exito",
                "se pudo crear el archivo" + ubicacion
            )
        else:
            QMessageBox.critical(
                self,
                "Error",
                "No se pudo crear el archivo" + ubicacion
            )




    @Slot()
    def click_mostrar(self):
        self.ui.salida.clear()
        self.ui.salida.insertPlainText(str(self.administrador))

    @Slot()
    def click_agregar_final(self):
        id = self.ui.id_lineEdit.text()
        origen_x = self.ui.origen_x_spinBox.value()
        origen_y = self.ui.origen_y_spinBox.value()
        destino_x = self.ui.destino_x_spinBox.value()
        destino_y = self.ui.destino_y_spinBox.value()
        velocidad = self.ui.velocidad_spinBox.value()
        red = self.ui.red_spinBox.value()
        green = self.ui.green_spinBox.value()
        blue = self.ui.blue_spinBox.value()
        distancia = self.ui.distancia_spinBox.value()

        particula=Particula(id,origen_x,origen_y,destino_x,destino_y,velocid
ad,red,green,blue,distancia)
        self.administrador.agregar_final(particula)

    @Slot()
    def click_agregar_inicio(self):
        id = self.ui.id_lineEdit.text()
        origen_x = self.ui.origen_x_spinBox.value()
        origen_y = self.ui.origen_y_spinBox.value()
        destino_x = self.ui.destino_x_spinBox.value()
        destino_y = self.ui.destino_y_spinBox.value()
        velocidad = self.ui.velocidad_spinBox.value()
        red = self.ui.red_spinBox.value()
        green = self.ui.green_spinBox.value()
        blue = self.ui.blue_spinBox.value()
        distancia = self.ui.distancia_spinBox.value()
```

```
        particula=Particula(id,origen_x,origen_y,destino_x,destino_y,velocid
ad,red,green,blue,distancia)
        self.administrador.agregar_inicio(particula)
```

administrador.py

```
from algoritmos import distancia_euclidiana, puntos_mas_cercanos
from particula import Particula
import json

class Adminisrador:
    def __init__(self):
        self.__particulas = []

    def agregar_final(self, particula: Particula):
        self.__particulas.append(particula)

    def agregar_inicio(self, particula: Particula):
        self.__particulas.insert(0, particula)

    def mostrar(self):
        for particula in self.__particulas:
            print(particula)

    def __str__(self):
        return "".join(
            str(particula)+'\n' for particula in self.__particulas
        )
    def guardar(self,ubicacion):
        try:
            with open(ubicacion, 'w') as archivo:
                lista = [particula.to_dict() for particula in
self.__particulas]
                print(lista)
                json.dump(lista, archivo, indent=5)
            return 1
        except :
            return 0

    def abrir(self, ubicacion):
        try:
            with open(ubicacion,'r') as archivo:
                lista=json.load(archivo)
                self.__particulas = [Particula(**particula) for particula in
lista]
            return 1
        except:
            return 0

    def __len__(self):
        return len(self.__particulas)
```

```
    def __iter__(self):
        self.cont=0

        return self

    def __next__(self):
        if self.cont < len(self.__particulas):
            particula = self.__particulas[self.cont]
            self.cont+=1
            return particula
        else:
            raise StopIteration
            #En cada metodo se vuelven a convertir a enteros y flotantes
para que haga bien la comparacion

    def ordenar_ID(self):  # Sirve para ordenar las particulas en orden
ascendente (ID)
        self.__particulas.sort(key=lambda particulas: int(particulas.id))

    # Sirve para ordenar las particulas en orden descendente (Distancia)
    def ordenar_DISTANCIA(self):
        self.__particulas.sort(key=lambda particulas: float(
            particulas.distancia), reverse=True)

    # Sirve para ordenar las particulas en orden ascendente (Velocidad)
    def ordenar_VELOCIDAD(self):
        self.__particulas.sort(
            key=lambda particulas: int(particulas.velocidad))

    def punto(self):
        p1 = []
        p2 = []
        for particula in self.__particulas:
            x1 = particula.origen_x
            y1 = particula.origen_y

            x2 = particula.destino_x
            y2 = particula.destino_y

            x=(x1,y1)
            y=(x2,y2)

            p1.append(x)
            p2.append(y)
        lista = p2+p1
        return puntos_mas_cercanos(lista)
```

particula.py

```
from algoritmos import distancia_euclidiana

class Particula:
    def __init__(self,
```

```
                id="", origen_x=0, origen_y=0, destino_x=0,
destino_y=0,velocidad=0,red=0,green=0,blue=0,distancia=0.0):

        self.__id = id
        self.__origen_x = origen_x
        self.__origen_y = origen_y
        self.__destino_x = destino_x
        self.__destino_y = destino_y
        self.__velocidad = velocidad
        self.__red = red
        self.__green = green
        self.__blue = blue
        self.__distancia = distancia_euclidiana(origen_x, origen_y,
destino_x, destino_y)



    def __str__(self):
        return (
            'Id: ' + str(self.__id) + '\n' +
            'Origen_x: ' + str(self.__origen_x) + '\n' +
            'Origen_y: ' + str(self.__origen_y) + '\n' +
            'Destino_x: ' + str(self.__destino_x) + '\n' +
            'Destino_y: ' + str(self.__destino_y) + '\n' +
            'Velocidad: ' + str(self.__velocidad) + '\n' +
            'Red: ' + str(self.__red) + '\n' +
            'Green: ' + str(self.__green) + '\n' +
            'Blue: ' + str(self.__blue) + '\n'+
            'Distancia: ' + str(self.__distancia) + '\n'
            )

    @property
    def id(self):
        return self.__id

    @property
    def origen_x(self):
        return self.__origen_x

    @property
    def origen_y(self):
        return self.__origen_y

    @property
    def destino_x(self):
        return self.__destino_x

    @property
    def destino_y(self):
        return self.__destino_y

    @property
    def velocidad(self):
```

```
        return self.__velocidad

    @property
    def red(self):
        return self.__red

    @property
    def green(self):
        return self.__green

    @property
    def blue(self):
        return self.__blue

    @property
    def distancia(self):
        return self.__distancia


    def to_dict(self):
        return {
            "id": self.__id,
            "origen_x": self.__origen_x,
            "origen_y": self.__origen_y,
            "destino_x": self.__destino_x,
            "destino_y": self.__destino_y,
            "velocidad": self.__velocidad,
            "red": self.__red,
            "green": self.__green,
            "blue": self.__blue
        }
```

algoritmos.py

```
from cmath import sqrt
import math


def distancia_euclidiana(x_1, y_1, x_2 ,y_2):

    return math.sqrt((x_2-x_1)**2 + (y_2-y_1)**2)


def puntos_mas_cercanos(puntos:list)->list:
    resultado=[]
    for punto_i in puntos:
        x1= punto_i[0]
        y1= punto_i[1]
        min=1000
        cercano=(0,0)
        for punto_j in puntos:
            if punto_i != punto_j:
                x2 = punto_j[0]
                y2 = punto_j[1]
```

```
                d=distancia_euclidiana(x1,y2,x2,y2)
                if d < min:
                    min=d
                    cercano=(x2,y2)
        resultado.append((punto_i, cercano))
    return resultado
```

ui_mainwindow.py

```
# -*- coding: utf-8 -*-

################################################################################
####
## Form generated from reading UI file 'mainwindow.ui'
##
## Created by: Qt User Interface Compiler version 5.15.2
##
## WARNING! All changes made in this file will be lost when recompiling UI
file!
################################################################################
####

from PySide2.QtCore import *
from PySide2.QtGui import *
from PySide2.QtWidgets import *


class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        if not MainWindow.objectName():
            MainWindow.setObjectName(u"MainWindow")
        MainWindow.resize(654, 552)
        MainWindow.setWindowOpacity(500.000000000000000)
        self.actionAbrir = QAction(MainWindow)
        self.actionAbrir.setObjectName(u"actionAbrir")
        self.actionGuardar = QAction(MainWindow)
        self.actionGuardar.setObjectName(u"actionGuardar")
        self.centralwidget = QWidget(MainWindow)
        self.centralwidget.setObjectName(u"centralwidget")
        self.gridLayout_3 = QGridLayout(self.centralwidget)
        self.gridLayout_3.setObjectName(u"gridLayout_3")
        self.tabWidget = QTabWidget(self.centralwidget)
        self.tabWidget.setObjectName(u"tabWidget")
        self.tab = QWidget()
        self.tab.setObjectName(u"tab")
        self.gridLayout_2 = QGridLayout(self.tab)
        self.gridLayout_2.setObjectName(u"gridLayout_2")
        self.groupBox = QGroupBox(self.tab)
        self.groupBox.setObjectName(u"groupBox")
        self.gridLayout = QGridLayout(self.groupBox)
        self.gridLayout.setObjectName(u"gridLayout")
        self.blue_spinBox = QSpinBox(self.groupBox)
        self.blue_spinBox.setObjectName(u"blue_spinBox")
        self.blue_spinBox.setMaximum(255)
```

```
        self.gridLayout.addWidget(self.blue_spinBox, 8, 1, 1, 1)

        self.label_6 = QLabel(self.groupBox)
        self.label_6.setObjectName(u"label_6")

        self.gridLayout.addWidget(self.label_6, 8, 0, 1, 1)

        self.agregar_final_pushButton = QPushButton(self.groupBox)
        self.agregar_final_pushButton.setObjectName(u"agregar_final_pushButt
on")

        self.gridLayout.addWidget(self.agregar_final_pushButton, 10, 0, 1,
2)

        self.origen_y_spinBox = QSpinBox(self.groupBox)
        self.origen_y_spinBox.setObjectName(u"origen_y_spinBox")
        self.origen_y_spinBox.setMaximum(500)

        self.gridLayout.addWidget(self.origen_y_spinBox, 2, 1, 1, 1)

        self.id_lineEdit = QLineEdit(self.groupBox)
        self.id_lineEdit.setObjectName(u"id_lineEdit")

        self.gridLayout.addWidget(self.id_lineEdit, 0, 1, 1, 1)

        self.label_7 = QLabel(self.groupBox)
        self.label_7.setObjectName(u"label_7")

        self.gridLayout.addWidget(self.label_7, 1, 0, 1, 1)

        self.distancia_spinBox = QSpinBox(self.groupBox)
        self.distancia_spinBox.setObjectName(u"distancia_spinBox")

        self.gridLayout.addWidget(self.distancia_spinBox, 9, 1, 1, 1)

        self.label_8 = QLabel(self.groupBox)
        self.label_8.setObjectName(u"label_8")

        self.gridLayout.addWidget(self.label_8, 2, 0, 1, 1)

        self.origen_x_spinBox = QSpinBox(self.groupBox)
        self.origen_x_spinBox.setObjectName(u"origen_x_spinBox")
        self.origen_x_spinBox.setMaximum(500)

        self.gridLayout.addWidget(self.origen_x_spinBox, 1, 1, 1, 1)

        self.velocidad_spinBox = QSpinBox(self.groupBox)
        self.velocidad_spinBox.setObjectName(u"velocidad_spinBox")
        self.velocidad_spinBox.setMaximum(2000)

        self.gridLayout.addWidget(self.velocidad_spinBox, 5, 1, 1, 1)
```

```
        self.label = QLabel(self.groupBox)
        self.label.setObjectName(u"label")

        self.gridLayout.addWidget(self.label, 3, 0, 1, 1)

        self.green_spinBox = QSpinBox(self.groupBox)
        self.green_spinBox.setObjectName(u"green_spinBox")
        self.green_spinBox.setMaximum(255)

        self.gridLayout.addWidget(self.green_spinBox, 7, 1, 1, 1)

        self.label_3 = QLabel(self.groupBox)
        self.label_3.setObjectName(u"label_3")

        self.gridLayout.addWidget(self.label_3, 5, 0, 1, 1)

        self.red_spinBox = QSpinBox(self.groupBox)
        self.red_spinBox.setObjectName(u"red_spinBox")
        self.red_spinBox.setMaximum(255)

        self.gridLayout.addWidget(self.red_spinBox, 6, 1, 1, 1)

        self.label_4 = QLabel(self.groupBox)
        self.label_4.setObjectName(u"label_4")

        self.gridLayout.addWidget(self.label_4, 6, 0, 1, 1)

        self.label_5 = QLabel(self.groupBox)
        self.label_5.setObjectName(u"label_5")

        self.gridLayout.addWidget(self.label_5, 7, 0, 1, 1)

        self.label_9 = QLabel(self.groupBox)
        self.label_9.setObjectName(u"label_9")

        self.gridLayout.addWidget(self.label_9, 9, 0, 1, 1)

        self.plainText_id_pushButton = QPushButton(self.groupBox)
        self.plainText_id_pushButton.setObjectName(u"plainText_id_pushButton
")

        self.gridLayout.addWidget(self.plainText_id_pushButton, 13, 0, 1, 2)

        self.label_2 = QLabel(self.groupBox)
        self.label_2.setObjectName(u"label_2")

        self.gridLayout.addWidget(self.label_2, 4, 0, 1, 1)

        self.plainText_distancia_pushButton = QPushButton(self.groupBox)
        self.plainText_distancia_pushButton.setObjectName(u"plainText_distan
cia_pushButton")
```

```
        self.gridLayout.addWidget(self.plainText_distancia_pushButton, 14,
0, 1, 2)

        self.destino_x_spinBox = QSpinBox(self.groupBox)
        self.destino_x_spinBox.setObjectName(u"destino_x_spinBox")
        self.destino_x_spinBox.setMaximum(500)

        self.gridLayout.addWidget(self.destino_x_spinBox, 3, 1, 1, 1)

        self.agregar_inicio_pushButton = QPushButton(self.groupBox)
        self.agregar_inicio_pushButton.setObjectName(u"agregar_inicio_pushBu
tton")

        self.gridLayout.addWidget(self.agregar_inicio_pushButton, 11, 0, 1,
2)

        self.mostar_pushButton = QPushButton(self.groupBox)
        self.mostar_pushButton.setObjectName(u"mostar_pushButton")

        self.gridLayout.addWidget(self.mostar_pushButton, 12, 0, 1, 2)

        self.label_10 = QLabel(self.groupBox)
        self.label_10.setObjectName(u"label_10")

        self.gridLayout.addWidget(self.label_10, 0, 0, 1, 1)

        self.destino_y_spinBox = QSpinBox(self.groupBox)
        self.destino_y_spinBox.setObjectName(u"destino_y_spinBox")
        self.destino_y_spinBox.setMaximum(500)

        self.gridLayout.addWidget(self.destino_y_spinBox, 4, 1, 1, 1)

        self.plainText_velocidad_pushButton = QPushButton(self.groupBox)
        self.plainText_velocidad_pushButton.setObjectName(u"plainText_veloci
dad_pushButton")

        self.gridLayout.addWidget(self.plainText_velocidad_pushButton, 15,
0, 1, 2)


        self.gridLayout_2.addWidget(self.groupBox, 0, 0, 1, 1)

        self.salida = QPlainTextEdit(self.tab)
        self.salida.setObjectName(u"salida")

        self.gridLayout_2.addWidget(self.salida, 0, 1, 1, 1)

        self.tabWidget.addTab(self.tab, "")
        self.tab_2 = QWidget()
        self.tab_2.setObjectName(u"tab_2")
        self.gridLayout_4 = QGridLayout(self.tab_2)
```

```
        self.gridLayout_4.setObjectName(u"gridLayout_4")
        self.buscar_lineEdit = QLineEdit(self.tab_2)
        self.buscar_lineEdit.setObjectName(u"buscar_lineEdit")

        self.gridLayout_4.addWidget(self.buscar_lineEdit, 1, 0, 1, 1)

        self.table_id_pushButton = QPushButton(self.tab_2)
        self.table_id_pushButton.setObjectName(u"table_id_pushButton")

        self.gridLayout_4.addWidget(self.table_id_pushButton, 1, 2, 1, 1)

        self.buscar_pushButton = QPushButton(self.tab_2)
        self.buscar_pushButton.setObjectName(u"buscar_pushButton")

        self.gridLayout_4.addWidget(self.buscar_pushButton, 1, 1, 1, 1)

        self.table_distancia_pushButton = QPushButton(self.tab_2)
        self.table_distancia_pushButton.setObjectName(u"table_distancia_push
Button")

        self.gridLayout_4.addWidget(self.table_distancia_pushButton, 1, 3,
1, 1)

        self.mostrar_tabla_pushButton = QPushButton(self.tab_2)
        self.mostrar_tabla_pushButton.setObjectName(u"mostrar_tabla_pushButt
on")

        self.gridLayout_4.addWidget(self.mostrar_tabla_pushButton, 1, 5, 1,
1)

        self.tabla = QTableWidget(self.tab_2)
        self.tabla.setObjectName(u"tabla")

        self.gridLayout_4.addWidget(self.tabla, 0, 0, 1, 6)

        self.table_velocidad_pushButton = QPushButton(self.tab_2)
        self.table_velocidad_pushButton.setObjectName(u"table_velocidad_push
Button")

        self.gridLayout_4.addWidget(self.table_velocidad_pushButton, 1, 4,
1, 1)

        self.tabWidget.addTab(self.tab_2, "")
        self.tab_3 = QWidget()
        self.tab_3.setObjectName(u"tab_3")
        self.gridLayout_5 = QGridLayout(self.tab_3)
        self.gridLayout_5.setObjectName(u"gridLayout_5")
        self.mostrar_puntos_pushButton = QPushButton(self.tab_3)
        self.mostrar_puntos_pushButton.setObjectName(u"mostrar_puntos_pushBu
tton")
```

```
        self.gridLayout_5.addWidget(self.mostrar_puntos_pushButton, 1, 1, 1,
1)

        self.limpiar_pushButton = QPushButton(self.tab_3)
        self.limpiar_pushButton.setObjectName(u"limpiar_pushButton")

        self.gridLayout_5.addWidget(self.limpiar_pushButton, 1, 3, 1, 1)

        self.dibujar_pushButton = QPushButton(self.tab_3)
        self.dibujar_pushButton.setObjectName(u"dibujar_pushButton")

        self.gridLayout_5.addWidget(self.dibujar_pushButton, 1, 0, 1, 1)

        self.graphicsView = QGraphicsView(self.tab_3)
        self.graphicsView.setObjectName(u"graphicsView")

        self.gridLayout_5.addWidget(self.graphicsView, 0, 0, 1, 4)

        self.dibujar_puntos_pushButton = QPushButton(self.tab_3)
        self.dibujar_puntos_pushButton.setObjectName(u"dibujar_puntos_pushBu
tton")

        self.gridLayout_5.addWidget(self.dibujar_puntos_pushButton, 1, 2, 1,
1)

        self.tabWidget.addTab(self.tab_3, "")

        self.gridLayout_3.addWidget(self.tabWidget, 0, 0, 1, 1)

        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QMenuBar(MainWindow)
        self.menubar.setObjectName(u"menubar")
        self.menubar.setGeometry(QRect(0, 0, 654, 21))
        self.menuArchivo = QMenu(self.menubar)
        self.menuArchivo.setObjectName(u"menuArchivo")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QStatusBar(MainWindow)
        self.statusbar.setObjectName(u"statusbar")
        MainWindow.setStatusBar(self.statusbar)

        self.menubar.addAction(self.menuArchivo.menuAction())
        self.menuArchivo.addAction(self.actionAbrir)
        self.menuArchivo.addAction(self.actionGuardar)

        self.retranslateUi(MainWindow)

        self.tabWidget.setCurrentIndex(2)


        QMetaObject.connectSlotsByName(MainWindow)
    # setupUi
```

```
    def retranslateUi(self, MainWindow):
        MainWindow.setWindowTitle(QCoreApplication.translate("MainWindow",
u"MainWindow", None))
        self.actionAbrir.setText(QCoreApplication.translate("MainWindow",
u"Abrir", None))
#if QT_CONFIG(shortcut)
        self.actionAbrir.setShortcut(QCoreApplication.translate("MainWindow"
, u"Ctrl+R", None))
#endif // QT_CONFIG(shortcut)
        self.actionGuardar.setText(QCoreApplication.translate("MainWindow",
u"Guardar", None))
#if QT_CONFIG(shortcut)
        self.actionGuardar.setShortcut(QCoreApplication.translate("MainWindo
w", u"Ctrl+S", None))
#endif // QT_CONFIG(shortcut)
        self.groupBox.setTitle("")
        self.label_6.setText(QCoreApplication.translate("MainWindow",
u"blue", None))
        self.agregar_final_pushButton.setText(QCoreApplication.translate("Ma
inWindow", u"Agregar Final", None))
        self.label_7.setText(QCoreApplication.translate("MainWindow",
u"origen x", None))
        self.label_8.setText(QCoreApplication.translate("MainWindow",
u"origen y", None))
        self.label.setText(QCoreApplication.translate("MainWindow",
u"destino x ", None))
        self.label_3.setText(QCoreApplication.translate("MainWindow",
u"velocidad", None))
        self.label_4.setText(QCoreApplication.translate("MainWindow",
u"red", None))
        self.label_5.setText(QCoreApplication.translate("MainWindow",
u"green", None))
        self.label_9.setText(QCoreApplication.translate("MainWindow",
u"distancia", None))
        self.plainText_id_pushButton.setText(QCoreApplication.translate("Mai
nWindow", u"ID (ASENDENTE)", None))
        self.label_2.setText(QCoreApplication.translate("MainWindow",
u"destino y", None))
        self.plainText_distancia_pushButton.setText(QCoreApplication.transla
te("MainWindow", u"DISTANCIA (DESCENDENTE)", None))
        self.agregar_inicio_pushButton.setText(QCoreApplication.translate("M
ainWindow", u"Agregar Inicio", None))
        self.mostar_pushButton.setText(QCoreApplication.translate("MainWindo
w", u"Mostrar", None))
        self.label_10.setText(QCoreApplication.translate("MainWindow",
u"id", None))
        self.plainText_velocidad_pushButton.setText(QCoreApplication.transla
te("MainWindow", u"VELOCIDAD (ASCENDENTE)", None))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab),
QCoreApplication.translate("MainWindow", u"Agregar", None))
        self.buscar_lineEdit.setPlaceholderText(QCoreApplication.translate("
MainWindow", u"ID", None))
        self.table_id_pushButton.setText(QCoreApplication.translate("MainWin
dow", u"ID (ASENDENTE)", None))
```

```
        self.buscar_pushButton.setText(QCoreApplication.translate("MainWindo
w", u"Buscar", None))
        self.table_distancia_pushButton.setText(QCoreApplication.translate("
MainWindow", u"DISTANCIA (DESCENDENTE)", None))
        self.mostrar_tabla_pushButton.setText(QCoreApplication.translate("Ma
inWindow", u"Mostrar", None))
        self.table_velocidad_pushButton.setText(QCoreApplication.translate("
MainWindow", u"VELOCIDAD (ASCENDENTE)", None))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab_2),
QCoreApplication.translate("MainWindow", u" Tabla", None))
        self.mostrar_puntos_pushButton.setText(QCoreApplication.translate("M
ainWindow", u"Mostrar puntos", None))
        self.limpiar_pushButton.setText(QCoreApplication.translate("MainWind
ow", u"Limpiar", None))
        self.dibujar_pushButton.setText(QCoreApplication.translate("MainWind
ow", u"Dibujar", None))
        self.dibujar_puntos_pushButton.setText(QCoreApplication.translate("M
ainWindow", u"Fuerza bruta", None))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab_3),
QCoreApplication.translate("MainWindow", u"Dibujo", None))
        self.menuArchivo.setTitle(QCoreApplication.translate("MainWindow",
u"Archivo", None))
    # retranslateUi
```