QScene

Cruz Collazo Wendy Paola

Seminario de Solución de Problemas de Algoritmia.

Lineamientos de Evaluación

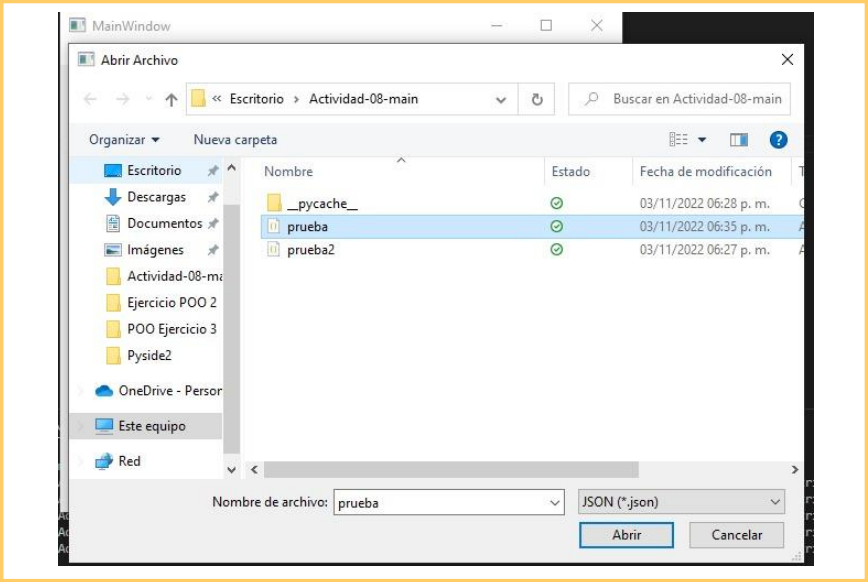
- *El reporte está en formato Google Docs o PDF.*
- *El reporte sigue las pautas del Formato de Actividades.*
- *El reporte tiene desarrollada todas las pautas del Formato de Actividades.*
- *Se muestra captura de pantalla de lo que se pide en el punto 2.*

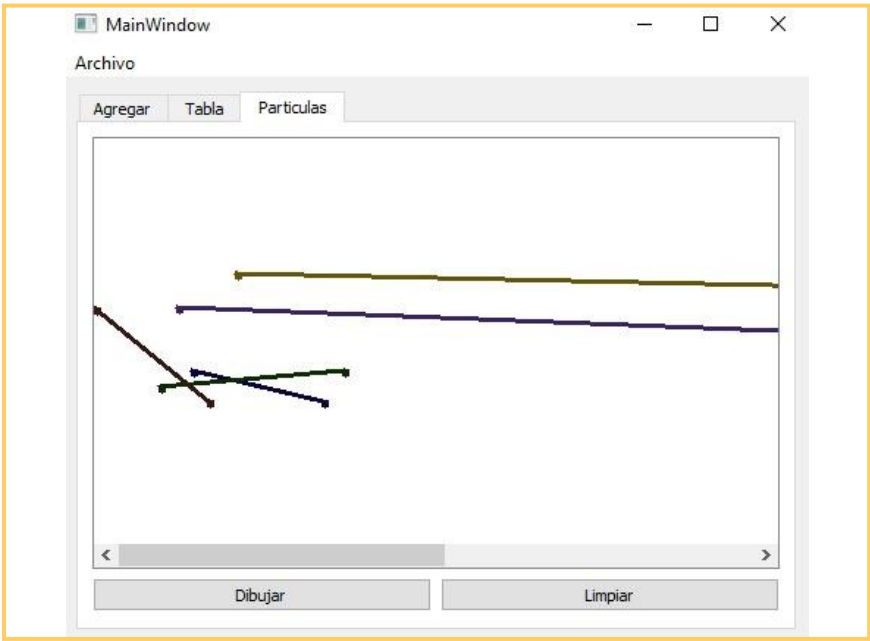
Desarrollo.

Comencé modificando el Desinger para agregar otro apartado donde poder dibujar las partículas.

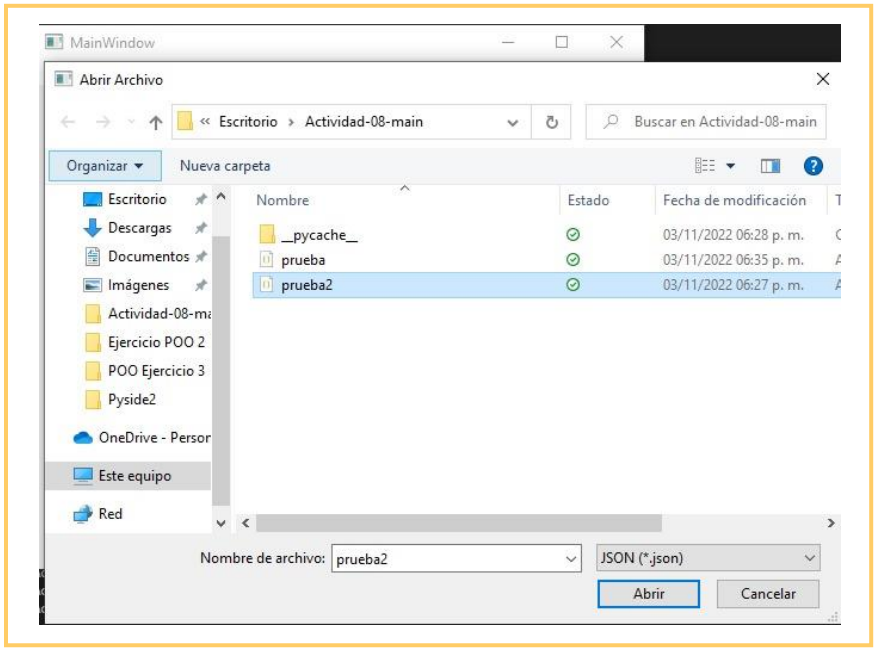
Actualicé el Mainwindow desde Python y comencé a implementar los botones de dibujar y limpiar.

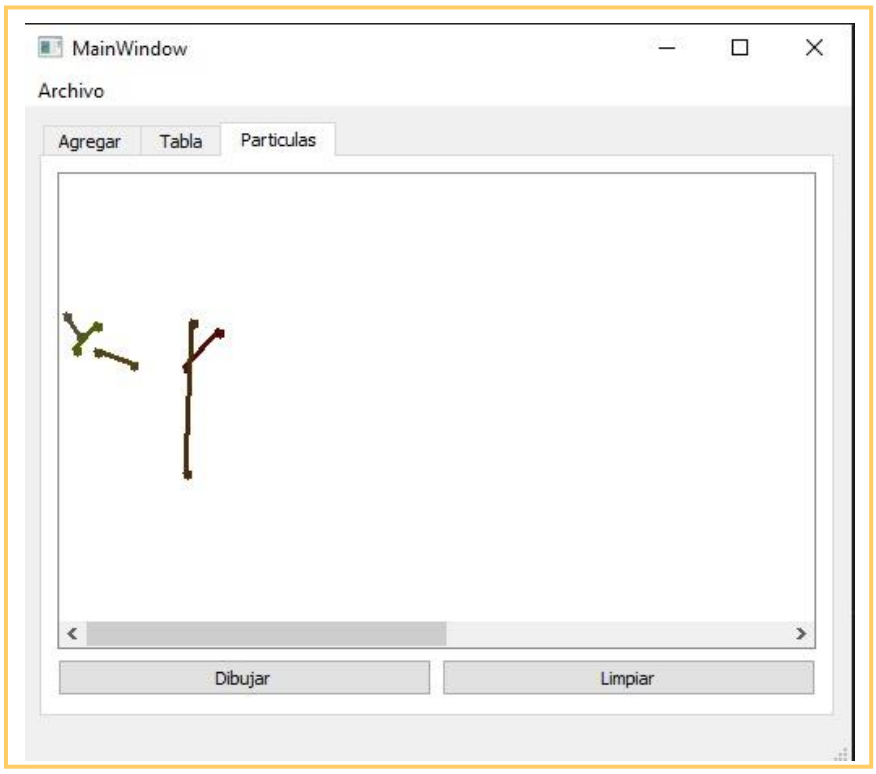
1) *Recupere el archivo .json donde tenia las partículas registradas, pero como salían puros puntos, guarde otro archivo con otras partículas.*





2)


MainWindow
Abrir Archivo
« Escritorio › Actividad-08-main
Buscar en Actividad-08-main
Organizar
Nueva carpeta
Escritorio
Descargas
Documentos
Imágenes
Actividad-08-ma
Ejercicio POO 2
POO Ejercicio 3
Pyside2
OneDrive - Persor
Este equipo
Red
Nombre
Estado
Fecha de modificación
__pycache__
prueba
prueba2
03/11/2022 06:28 p. m.
03/11/2022 06:35 p. m.
03/11/2022 06:27 p. m.
Nombre de archivo:
prueba2
JSON (*.json)
Abrir
Cancelar


MainWindow
Archivo
Agregar
Tabla
Particulas
Dibujar
Limpiar

Conclusión.

Fue una actividad muy fácil e interesante ya que no te puedes imaginar de todas las cosas que se pueden hacer con esta aplicación, para implementar los métodos fue sencillo ya que básicamente en dibujar fue casi lo mismo en agregar tabla.

Y para limpiar simplemente era poner algo y ya.

Referencias.

- ✓ *PySide2- QScene (Qt for Python (VI)) – Michel Davalos Boites.*
  https://www.youtube.com/watch?v=3jHTFzPpZY8

Código.

- *Administradora.py*

```
from particula import Particula
import json

class Administradora:
    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) for particula in
self.__particulas
            )

    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

    def guardar(self,ubiacion):
        try:
            with open(ubiacion,'w') as archivo:
                lista = [particula.to_dict() for particula
in self.__particulas]
                json.dump(lista,archivo, indent = 5)
            return
        except:
            return 0
             #json.dump()

    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
```

➢ *Algoritmos.py*

```
➢ import math
➢
➢ def distancia_euclidiana(x_1, y_1, x_2, y_2):
➢     a = (x_2 - x_1)*(x_2 - x_1)
➢     b = (y_2 - y_1)*(y_2 - y_1)
➢
➢     c = a + b
➢
➢     distancia = math.sqrt(c)
➢
➢     return distancia
➢
```

➢ *Main.py*

```
➢ from PySide2.QtWidgets import QApplication
➢ from mainwindow import MainWindow
➢ import sys
➢
➢ app = QApplication()
➢
➢ window = MainWindow()
➢
➢ window.show()
➢
➢ sys.exit(app.exec_())
```

➢ *Mainwindow.py*

```
➢
➢ from PySide2.QtWidgets import
  QMainWindow,QFileDialog,QMessageBox,QTableWidgetItem,
  QGraphicsScene
➢ from ui_mainwindow import Ui_MainWindow
➢ from administradora import Administradora
➢ from particula import Particula
➢ from PySide2.QtCore import Slot
➢ from PySide2.QtGui import QPen, QColor, QTransform
➢
➢ class MainWindow(QMainWindow):
➢     def __init__(self):
➢         super(MainWindow,self).__init__()
➢
➢         self.administrador = Administradora()
➢
```

```
        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)
        self.ui.Agregar_final.clicked.connect(self.agregar_final)
        self.ui.Agregar_Inicio.clicked.connect(self.agregar_inicio)
        self.ui.Mostrar.clicked.connect(self.ver)

        self.ui.actionAbrir.triggered.connect(self.action_abrir_archivo)
        self.ui.actionGuardar.triggered.connect(self.action_guardar_archivo)

        self.ui.view_button.clicked.connect(self.mostrar_tabla)
        self.ui.search_button.clicked.connect(self.buscar_tabla)

        self.ui.dibujar.clicked.connect(self.dibujar)
        self.ui.limpiar.clicked.connect(self.limipiar)

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

    @Slot()
    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 dibujar(self):
        pen = QPen()
        pen.setWidth(3)

        for particula in self.administrador:
            origenx = int(particula.origen_x)
            origeny = int(particula.origen_y)
            destinox = int(particula.destino_x)
            destinoy = int(particula.destino_y)

            red = int(particula.red)
            green = int(particula.green)
            blue = int(particula.blue)
```

```
➢
➢               color = QColor(red, green, blue)
➢               pen.setColor(color)
➢
➢               self.scene.addEllipse(origenx, origeny, 3, 3,
   pen)
➢               self.scene.addEllipse(destinox, destinoy, 3, 3,
   pen)
➢               self.scene.addLine(origenx, origeny, destinox,
   destinoy, pen)
➢
➢
➢       @Slot()
➢       def limipiar(self):
➢           self.scene.clear()
➢
➢
➢       @Slot()
➢       def buscar_tabla(self):
➢           id = self.ui.search_line.text()
➢           encontrado = False
➢
➢           for particula in self.administrador:
➢
➢               if int(id)  ==  particula.id:
➢                   self.ui.table.clear()
➢                   self.ui.table.setRowCount(1)
➢                   headers = ["ID","Origen X","Origen
   Y","Destino X","Destino Y","Red","Green","Blue","Distancia"]
➢                   self.ui.table.setHorizontalHeaderLabels(head
   ers)
➢
➢                   id_widget =
   QTableWidgetItem(str(particula.id))
➢                   origenx_widget =
   QTableWidgetItem(str(particula.origen_x))
➢                   origeny_widget =
   QTableWidgetItem(str(particula.origen_y))
➢                   destinox_widget =
   QTableWidgetItem(str(particula.destino_x))
➢                   destinoy_widget =
   QTableWidgetItem(str(particula.destino_y))
➢                   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.table.setItem(0,0,id_widget)
                self.ui.table.setItem(0,1,origenx_widget)
                self.ui.table.setItem(0,2,origeny_widget)
                self.ui.table.setItem(0,3,destinox_widget)
                self.ui.table.setItem(0,4,destinoy_widget)
                self.ui.table.setItem(0,5,red_widget)
                self.ui.table.setItem(0,6,green_widget)
                self.ui.table.setItem(0,7,blue_widget)
                self.ui.table.setItem(0,8,distancia_widget)

                encontrado = True

                return

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

    @Slot()
    def mostrar_tabla(self):
        self.ui.table.setColumnCount(9)
        headers = ["ID","Origen X","Origen Y","Destino
X","Destino Y","Red","Green","Blue","Distancia"]
        self.ui.table.setHorizontalHeaderLabels(headers)

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

        row = 0
        for particula in self.administrador:
            id_widget = QTableWidgetItem(str(particula.id))
            origenx_widget =
QTableWidgetItem(str(particula.origen_x))
            origeny_widget =
QTableWidgetItem(str(particula.origen_y))
            destinox_widget =
QTableWidgetItem(str(particula.destino_x))
            destinoy_widget =
QTableWidgetItem(str(particula.destino_y))
```

```
            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.table.setItem(row,0,id_widget)
            self.ui.table.setItem(row,1,origenx_widget)
            self.ui.table.setItem(row,2,origeny_widget)
            self.ui.table.setItem(row,3,destinox_widget)
            self.ui.table.setItem(row,4,destinoy_widget)
            self.ui.table.setItem(row,5,red_widget)
            self.ui.table.setItem(row,6,green_widget)
            self.ui.table.setItem(row,7,blue_widget)
            self.ui.table.setItem(row,8,distancia_widget)

            row += 1


    @Slot()
    def action_abrir_archivo(self):
        ubicacion = QFileDialog.getOpenFileName(self,'Abrir Archivo','.','JSON (*.json)')[0]
        if self.administrador.abrir(ubicacion):
            QMessageBox.information(self,"Exito","Se abrió el archivo de" + ubicacion)
        else:
            QMessageBox.information(self,"Error","No se pudo abrir el archivo de " + ubicacion)


    @Slot()
    def action_guardar_archivo(self):
        ubicacion = QFileDialog.getSaveFileName(self,'Guardar Archivo','.','JSON (*.json)')[0]
        if self.administrador.guardar(ubicacion):
            QMessageBox.information(self,"Exito","Se creó el archivo con exito en " + ubicacion)
        else:
            QMessageBox.information(self,"Error","No se pudo crear el archivo en " + ubicacion)
```

```
➢
➢
➢
➢      @Slot()
➢      def ver(self):
➢          self.ui.Print.clear()
➢          self.ui.Print.insertPlainText(str(self.administrador
   ))
➢
➢
➢
➢      @Slot()
➢      def agregar_final(self):
➢          ID = self.ui.ID_spinBox.value()
➢          OrigenX = self.ui.OrigenX_spinBox.value()
➢          OrigenY = self.ui.OrigenY_spinBox.value()
➢          DestinoX = self.ui.DestinoX_spinBox.value()
➢          DestinoY = self.ui.DestinoY_spinBox.value()
➢          Red = self.ui.Red_spinBox.value()
➢          Green = self.ui.Green_spinBox.value()
➢          Blue = self.ui.Blue_spinBox.value()
➢
➢
➢          particula1 =
   Particula(ID,OrigenX,OrigenY,DestinoX,DestinoY,Red,Green,Blu
   e)
➢          self.administrador.agregar_final(particula1)
➢
➢      @Slot()
➢      def agregar_inicio(self):
➢          ID = self.ui.ID_spinBox.value()
➢          OrigenX = self.ui.OrigenX_spinBox.value()
➢          OrigenY = self.ui.OrigenY_spinBox.value()
➢          DestinoX = self.ui.DestinoX_spinBox.value()
➢          DestinoY = self.ui.DestinoY_spinBox.value()
➢          Red = self.ui.Red_spinBox.value()
➢          Green = self.ui.Green_spinBox.value()
➢          Blue = self.ui.Blue_spinBox.value()
➢
➢          particula1 =
   Particula(ID,OrigenX,OrigenY,DestinoX,DestinoY,Red,Green,Blu
   e)
➢          self.administrador.agregar_inicio(particula1)
```

➢ *particula.py*

```
➢ from algoritmos import distancia_euclidiana
➢
➢ class Particula:
➢     def __init__(self,id = 0, origen_x = 0, origen_y = 0,
  destino_x = 0, destino_y=0,red = 0, green = 0, blue = 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.__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 en
  X :' + str(self.__origen_x) + '\n' +
➢                'Origen en Y :' + str(self.__origen_y) + '\n'
  + 'Destino en X :' + str(self.__destino_x) + '\n' +
➢                'Destino en Y: ' + str(self.__destino_y) +
  '\n' + 'Distancia : ' + str(self.__distancia) + '\n' +
➢                'Red :' + str(self.__red) + '\n' 'Green :' +
  str(self.__green) + '\n' 'Blue :' + str(self.__blue) + '\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 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,
            "red": self.__red,
            "green": self.__green,
            "blue": self.__blue
        }
```