Actividad 09 // QScene

Sámano Juárez Juan Jesus.

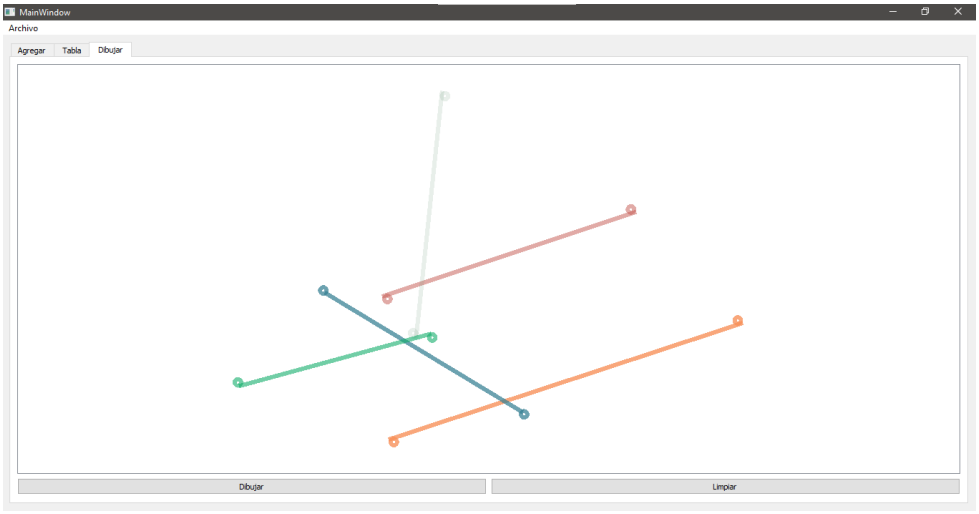
Seminario de Solución de Problemas de Algoritmia

Lineamiento de evaluación.

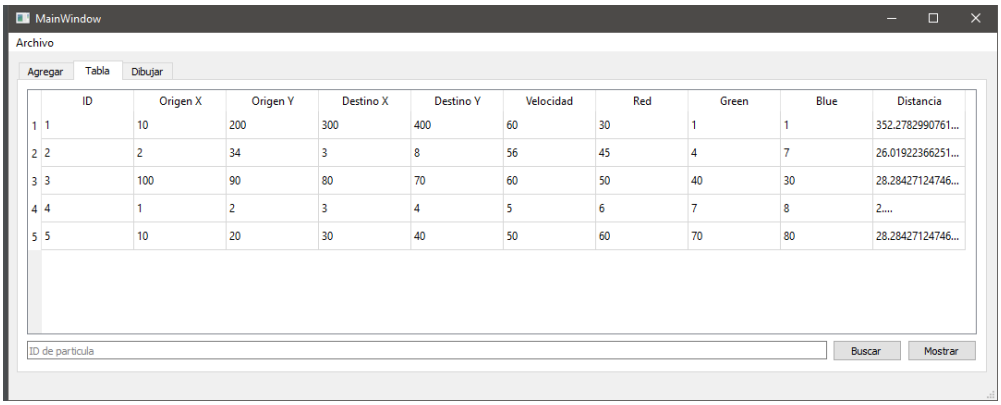
- [ ] El reporte está en formato Google Docs o PDF.
- [ ] El reporte sigue las pautas del Formato de Actividades .
- [ ] El reporte tiene desarrollada todas las pautas del Formato de Actividades.
- [ ] Se muestra captura de pantalla de lo que se pide en el punto 2.

Desarrollo.

Toma capturas de pantalla de la ejecución mostrando la visualización de al menos 5 partículas en el QScene.



5 partículas en tabla.



	ID	Origen X	Origen Y	Destino X	Destino Y	Velocidad	Red	Green	Blue	Distancia
1	1	10	200	300	400	60	30	1	1	352.2782990761...
2	2	2	34	3	8	56	45	4	7	26.01922366251...
3	3	100	90	80	70	60	50	40	30	28.28427124746...
4	4	1	2	3	4	5	6	7	8	2....
5	5	10	20	30	40	50	60	70	80	28.28427124746...

Conclusiones.

En esta actividad solo tuve la una complicación al colocar los colores a cada grafica de las partículas, todo lo demas fue muy sencillo siguiendo los pasos del video proporcionado por el profesor.

Referencias.

Michel Davalos Boites.[MICHEL DAVALOS BOITES](04/10/2022) PySide2 – Qscene(Qt for Python)(VI)[Archivo de video].
https://www.youtube.com/watch?v=3jHTFzPpZY8&ab_channel=MICHELDAVALOSBOITES

Código.

main.py

```
 from PySide2.QtWidgets import QPushButton, QApplication

from mainwindow import MainWindow
import sys

#Aplicación de QT
app = QApplication()
#Crear objeto
window = MainWindow()
#Hacer visible el elemento Botón
window.show()
#Qt loop
sys.exit(app.exec_())
```

mainwindow.py

```
from ast import Str
from math import fabs
from multiprocessing import managers
from sqlite3 import Row
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 manager import Manager
from particula import Particula
from random import randint

class MainWindow(QMainWindow):
    def __init__(self):
        super(MainWindow, self).__init__()
        self.manager = Manager()
        self.ui = Ui_MainWindow()
```

```
        self.ui.setupUi(self)
        self.id = 0

        #Cuando el botón pushbutton es presionado, ejecuta la función
click_agregar
        # self.ui.mostrar.clicked.connect(self.click_mostrar)
        self.ui.insertar_inicio.clicked.connect(self.click_insertar_inicio)
        self.ui.insertar_final.clicked.connect(self.click_insertar_final)
        self.ui.mostrar.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_titulo)
        self.ui.dibujar.clicked.connect(self.dibujar)
        self.ui.limpiar.clicked.connect(self.limpiar)

        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 dibujar(self):
        pen = QPen()
        pen.setWidth(2)

        for particula in self.manager:
            origen_x = randint (0, 255)
            origen_y = randint (0, 255)
            destino_x = randint(0,255)
            destino_y = randint(0,255)

            color = QColor(origen_x, origen_y, destino_x, destino_y)
            pen.setColor(color)


            self.scene.addEllipse(origen_x, origen_y, 3, 3, pen)
            self.scene.addEllipse(destino_x, destino_y, 3, 3,  pen)
```

```
            self.scene.addLine(origen_x+3, origen_y+3, destino_x,
destino_y,  pen)


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


    @Slot()
    def buscar_titulo(self):
        id = self.ui.buscar_lineEdit.text() #str
        encontrado = False

        for particula in self.manager:
            if str (id) == str(particula.id):
                self.ui.tabla.clear()
                self.ui.tabla.setRowCount(1)

                id_widget = QTableWidgetItem(str(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 nombre "{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.manager))

    row = 0
    for particula in self.manager:
        id_widget = QTableWidgetItem(str(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
```

```python
#Funcion que es llamada por x razón que imprime Click en Terminal.
@Slot()
# def click_mostrar(self):
#     a

@Slot()
def action_abrir_archivo(self):
    #print("Abrir_archivo")
    ubicacion = QFileDialog.getOpenFileName(
        self,
        'Abrir Archivo',
        '.',
        'JSON (*.json)'
    ) [0]
    if self.manager.abrir(ubicacion):
        QMessageBox.information(
            self,
            "Éxito",
            "Se abrió 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 Archivo',
        '.',
        'JSON (*.json)'
    )[0]
    print(ubicacion)
    if self.manager.guardar(ubicacion):
        QMessageBox.information(
            self,
            "Exito",
            "Se pudo crear el archivi" + ubicacion
        )
```

```
        else:
            QMessageBox.critical(
                self,
                "Error",
                "No se pudo crear el archivo" + ubicacion
            )




    def click_insertar_inicio(self):
        self.id += 1
        aux = Particula(self.id, self.ui.ox.value(), self.ui.oy.value(),
self.ui.dx.value(), self.ui.dy.value(), self.ui.velocidad.value(),
self.ui.red.value(), self.ui.green.value(), self.ui.blue.value())
        self.manager.agregarInicio(aux)
        self.click_mostrar()

    def click_insertar_final(self):
        self.id += 1
        aux = Particula(self.id , self.ui.ox.value(), self.ui.oy.value(),
self.ui.dx.value(), self.ui.dy.value(), self.ui.velocidad.value(),
self.ui.red.value(), self.ui.green.value(), self.ui.blue.value())
        self.manager.agregarFinal(aux)
        self.click_mostrar()

    def click_mostrar(self):
        self.ui.lista_particulas.clear()
        self.ui.lista_particulas.insertPlainText(str(self.manager))
```