Actividad 09 - QSence

Gabriel Eduardo Sevilla Chavez

Seminario de algoritmia

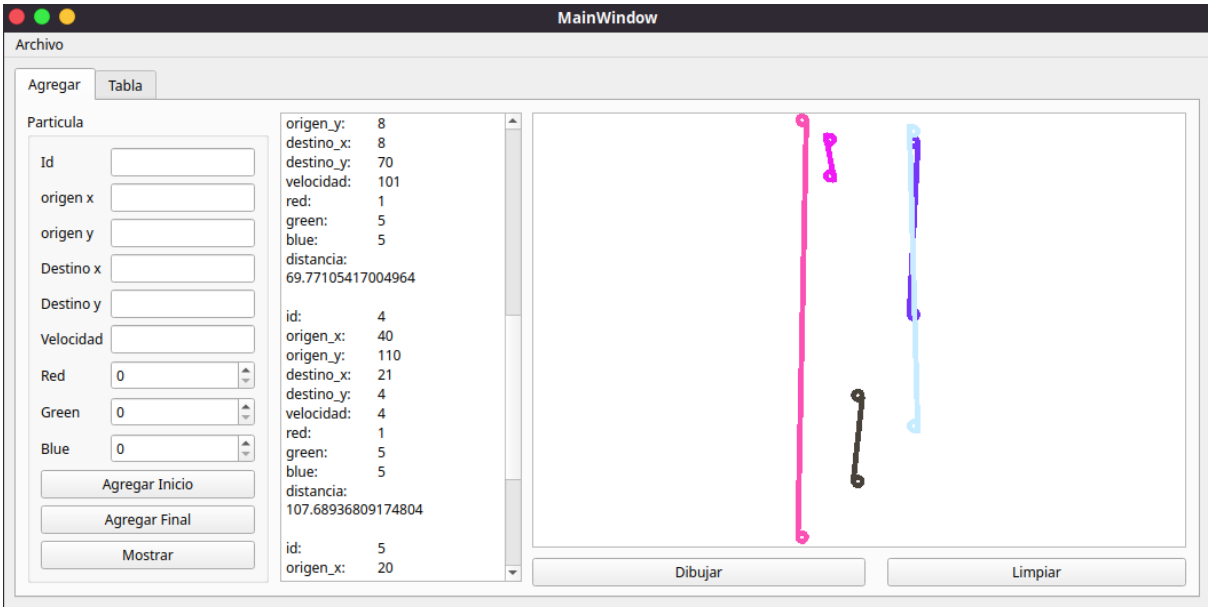
Lineamientos de evaluación

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

visualización de al menos 5 partículas en el `QScene`.



Conclusiones

Fue interesante esta actividad, ya que, aparte de gráficas las partículas en un espacio con todo y sus colores (cosa que nunca había hecho), fue el ingeniartelas o pensar un poco en trasladar lo del video de referencia a dicha actividad.

Referencias

MICHEL DAVALOS BOITES. (2020b, noviembre 5). *PySide2 - QScene (Qt for Python)(VI)*. YouTube. https://www.youtube.com/watch?v=3jHTFzPpZY8

Código

main.py

```
from PySide2.QtWidgets import QApplication

from mainWindow import MainWindow

import sys


app = QApplication()

#Ventana de app

window = MainWindow()

window.show()

sys.exit(app.exec_())
```

mainWindow.py

```
from PySide2.QtWidgets import QMainWindow, QFileDialog, QMessageBox, QTableWidgetItem, QGraphicsScene

from PySide2.QtGui import QPen,QColor,QTransform

from PySide2.QtCore import Slot

from ui_mainWindow import Ui_MainWindow

from random import  randint

#incluir clases particlas
```

```
from Actividad05_ClasesyObjetos.particulas import Particulas
from Actividad05_ClasesyObjetos.particula import Particula
class MainWindow(QMainWindow):
   def __init__(self):
        super(MainWindow, self).__init__() #Contructor de QMainWindow
        #Guardar particulas
        self.particulas = Particulas()

        self.ui = Ui_MainWindow()
        #mandar los datos de self.ui a la ventana
        self.ui.setupUi(self)
        # Eventos en botones

self.ui.pbAgregarInicio.clicked.connect(self.click_agregarInicio)

self.ui.pbAgregaFinal.clicked.connect(self.click_agregarFinal)
        self.ui.pbMostrar.clicked.connect(self.click_mostrar)
        #Ad Archivo

self.ui.actionAbrir_archivo.triggered.connect(self.actionOpenFile)

self.ui.actionGuardar_archivo.triggered.connect(self.actionSaveFile)
        #Table
        self.ui.btnMostrarTabla.clicked.connect(self.mostrarTabla)
```

```
        self.ui.btnBuscar.clicked.connect(self.buscarId)
        #Dibujo
        self.ui.btnDibujar.clicked.connect(self.dibujar)
        self.ui.btnLimpiar.clicked.connect(self.limpiar)
        #Escena
        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.particulas:
            r=randint(0,255)
```

```
            g=randint(0,255)
            b=randint(0,255)
            color = QColor(r,g,b)
            pen.setColor(color)
            origen_x  = particula.origen_x
            origen_y  = particula.origen_y
            destino_x = particula.destino_x
            destino_y = particula.destino_y
            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 buscarId(self):
        id = self.ui.lineEditTabla.text()
        encontrado = False
```

```
        for particula in self.particulas:
            if id == str(particula.id):
                self.ui.tableWidget.clear()
                self.ui.tableWidget.setRowCount(1)
                self.viewData(0,particula)
                encontrado = True
                return
        if not encontrado:
            QMessageBox.warning(
                self,"Atencion",f'El id "{id}" no se encuentra'
            )




    @Slot()
    def mostrarTabla(self):
        self.ui.tableWidget.setColumnCount(10)
        headers = ["id","origen_x","origen_y","destino_x",

"destino_y","velocidad","red","green","blue","distancia"]
        self.ui.tableWidget.setHorizontalHeaderLabels(headers)
        self.ui.tableWidget.setRowCount(len(self.particulas))
```

```
        row = 0
        for particula in self.particulas:
            self.viewData(row,particula)
            row +=1

    def viewData(self,row,particula):
        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.tableWidget.setItem(row,0,id_widget)
        self.ui.tableWidget.setItem(row,1,origen_x_widget)
        self.ui.tableWidget.setItem(row,2,origen_y_widget)
        self.ui.tableWidget.setItem(row,3,destino_x_widget)
        self.ui.tableWidget.setItem(row,4,destino_y_widget)
```

```
        self.ui.tableWidget.setItem(row,5,velocidad_widget)
        self.ui.tableWidget.setItem(row,6,red_widget)
        self.ui.tableWidget.setItem(row,7,green_widget)
        self.ui.tableWidget.setItem(row,8,blue_widget)
        self.ui.tableWidget.setItem(row,9,distancia_widget)

    @Slot()
    def actionOpenFile(self):
        ubicacion = QFileDialog.getOpenFileName(
            self,
            'Abrir archivo',
            '.',
            'JSON (*.json)'
        )[0]
        if self.particulas.abrir(ubicacion):
            QMessageBox.information(
                self,'Exito',
                'Se abrio el archivo'+ubicacion
            )
        else:
            QMessageBox.critical(
                self,'Error',
```

```
                'Error al abrir el archivo'+ubicacion
            )
    def actionSaveFile(self):
        #print("Guardar archivo")
        ubicacion = QFileDialog.getSaveFileName(
            self,
            'Guardar archivo',
            '.',
            'JSON (*.json)'
        )[0]
        print(ubicacion)
        if self.particulas.guardar(ubicacion):
            QMessageBox.information(
                self,"Exito","Archivo guardado"+ubicacion
            )
        else:
            QMessageBox.critical(
                self,"Error","No se pudo guardar el archivo"+
ubicacion
            )
    @Slot()
    def click_mostrar(self):
```

```
        self.ui.salida.clear()
        #self.particulas.mostrar()
        self.ui.salida.insertPlainText(str(self.particulas))

    @Slot() #Guardar los datos obenidos
    def click_agregarInicio(self):
        id = self.ui.leId.text()
        origenx = self.ui.leOrigenx.text()
        origeny = self.ui.leOrigenY.text()
        destinox = self.ui.leDestinoX.text()
        destinoy = self.ui.leDestinoY.text()
        velocidad = self.ui.leVelocidad.text()
        red = self.ui.sbRed.value()
        green = self.ui.sbGreen.value()
        blue = self.ui.sbBlue.value()
        #Crear particla
        particula =
Particula(int(id),int(origenx),int(origeny),int(destinox),int(destin
oy),int(velocidad),red,green,blue)
        self.particulas.agregar_inicio(particula)

    @Slot() #Guardar los datos obenidos
    def click_agregarFinal(self):
```

```
        id = self.ui.leId.text()

        origenx = self.ui.leOrigenx.text()

        origeny = self.ui.leOrigenY.text()

        destinox = self.ui.leDestinoX.text()

        destinoy = self.ui.leDestinoY.text()

        velocidad = self.ui.leVelocidad.text()

        red = self.ui.sbRed.value()

        green = self.ui.sbGreen.value()

        blue = self.ui.sbBlue.value()

        #Crear particla

        particulafinal =
Particula(int(id),int(origenx),int(origeny),int(destinox),int(destin
oy),int(velocidad),red,green,blue)

        self.particulas.agregar_final(particulafinal)
```