Actividad 07 - QFileDialog

Arturo Sánchez Sánchez

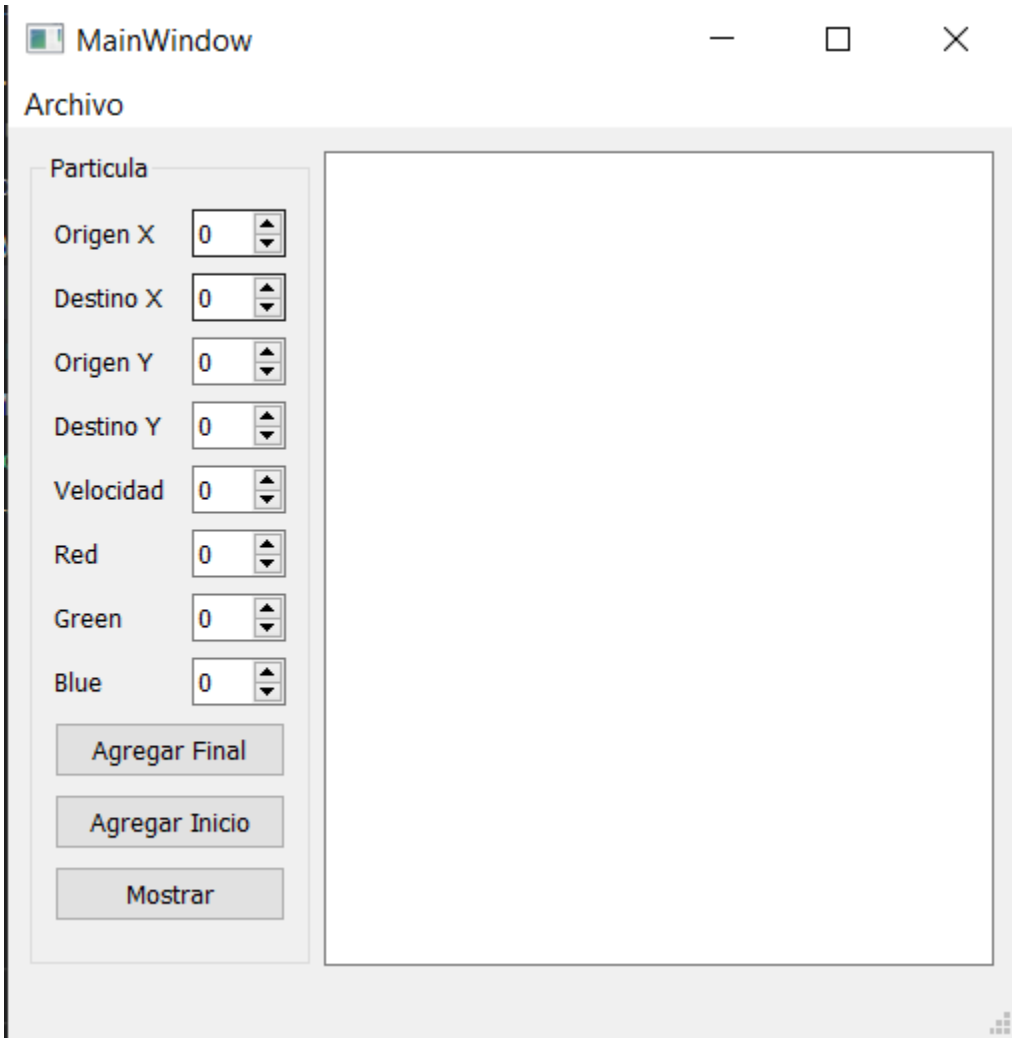
Seminario de Algoritmia

Lineamientos de evaluación

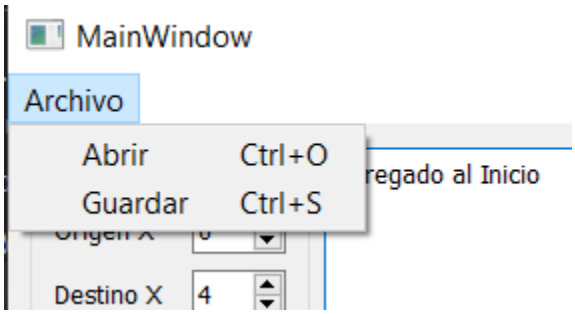
- El reporte está en formato Google Docs o PDF.
- El reporte sigue las pautas del Formato de Actividades .
- El reporte tiene desarrollada todas las pautas del Formato de Actividades.
- Se muestra la captura de pantalla de las partículas con el método mostrar() previo a generar el respaldo.
- Se muestran capturas de pantallas de los pasos que se realizan en la interfaz para generar el respaldo.
- Se muestra el contenido del archivo *.json.*
- Se muestran capturas de pantallas de los pasos que se realizan en la interfaz para abrir el archivo de respaldo *.json*.
- Se muestra la captura de pantalla de las partículas con el método mostrar() después de abrir el respaldo.

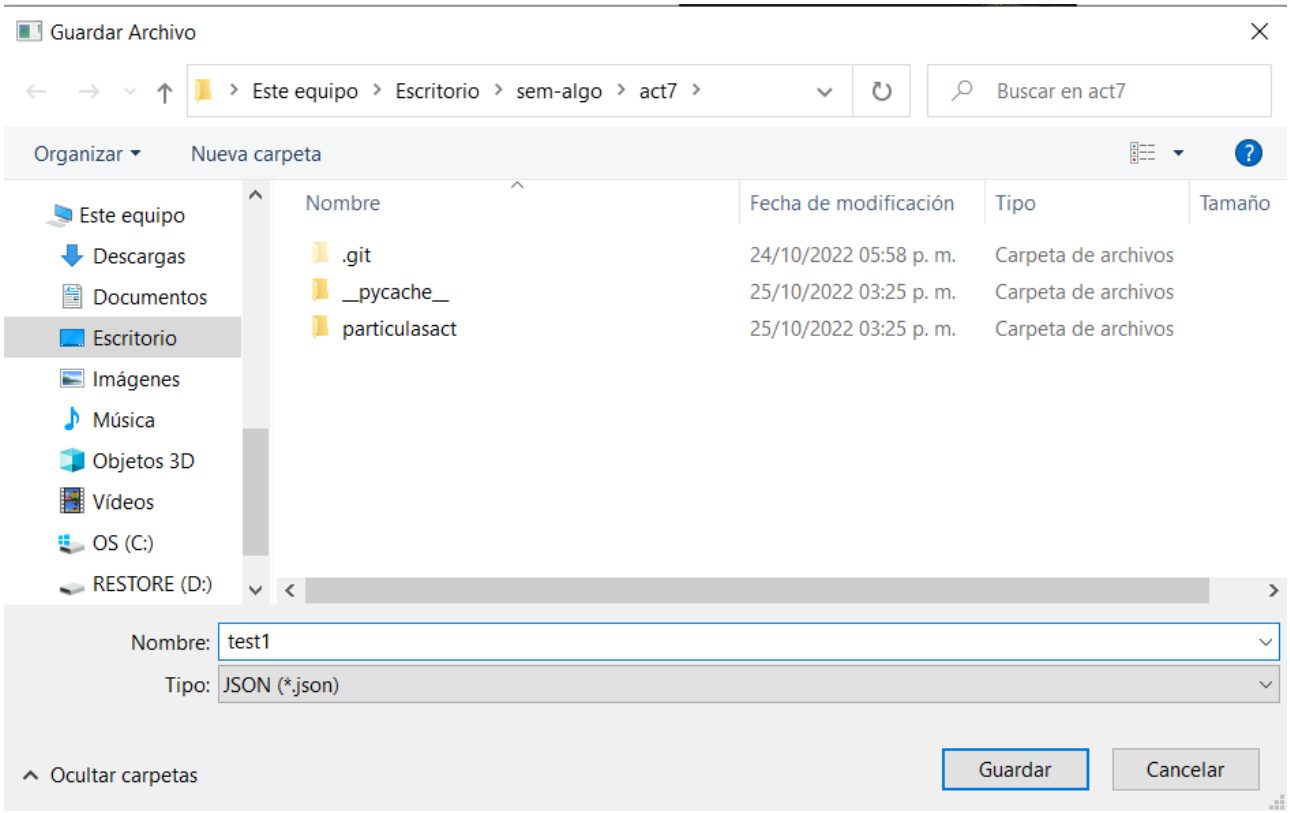
Desarrollo

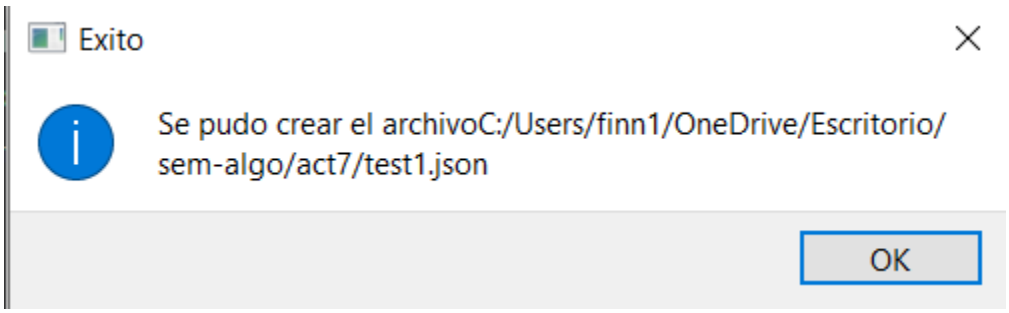
Captura de pantalla de las partículas con el método mostrar previo a generar el respaldo



Capturas de pantallas de los pasos que se realizan en la interfaz para generar el respaldo



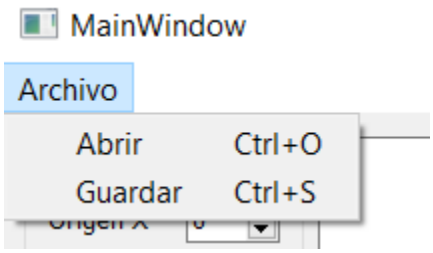


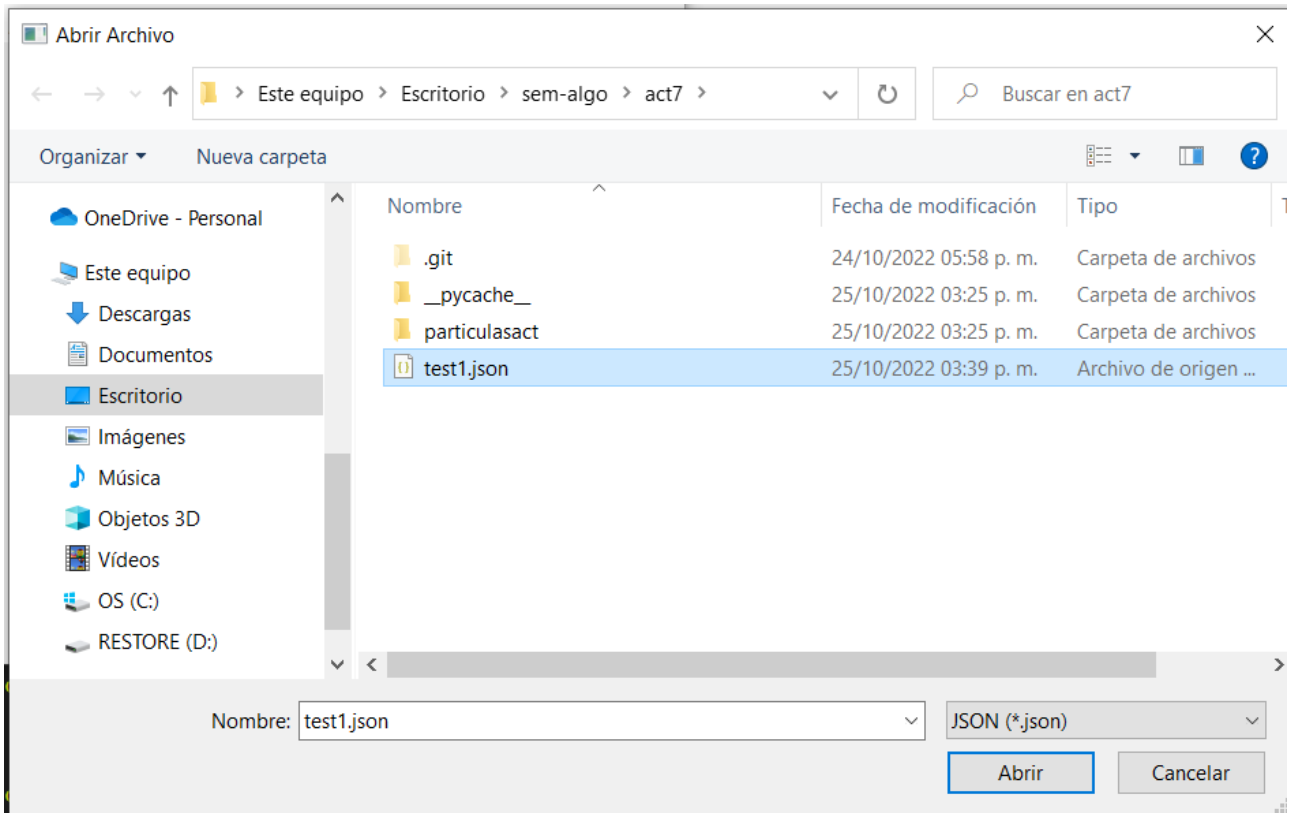


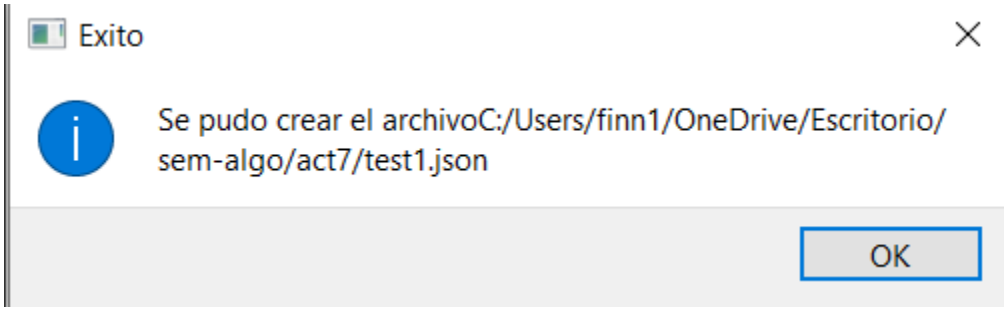
Muestra el contenido del archivo *.json*

```
[
 {
  "id": 2,
  "origen_x": 3,
  "origen_y": 2,
  "destino_x": 7,
  "destino_y": 10,
  "velocidad": 1,
  "red": 9,
  "green": 4,
  "blue": 5,
  "distancia": "8.94427"
 },
 {
  "id": 1,
  "origen_x": 1,
  "origen_y": 1,
  "destino_x": 3,
  "destino_y": 5,
  "velocidad": 1,
  "red": 9,
  "green": 4,
  "blue": 5,
  "distancia": "4.47214"
 }
]
```

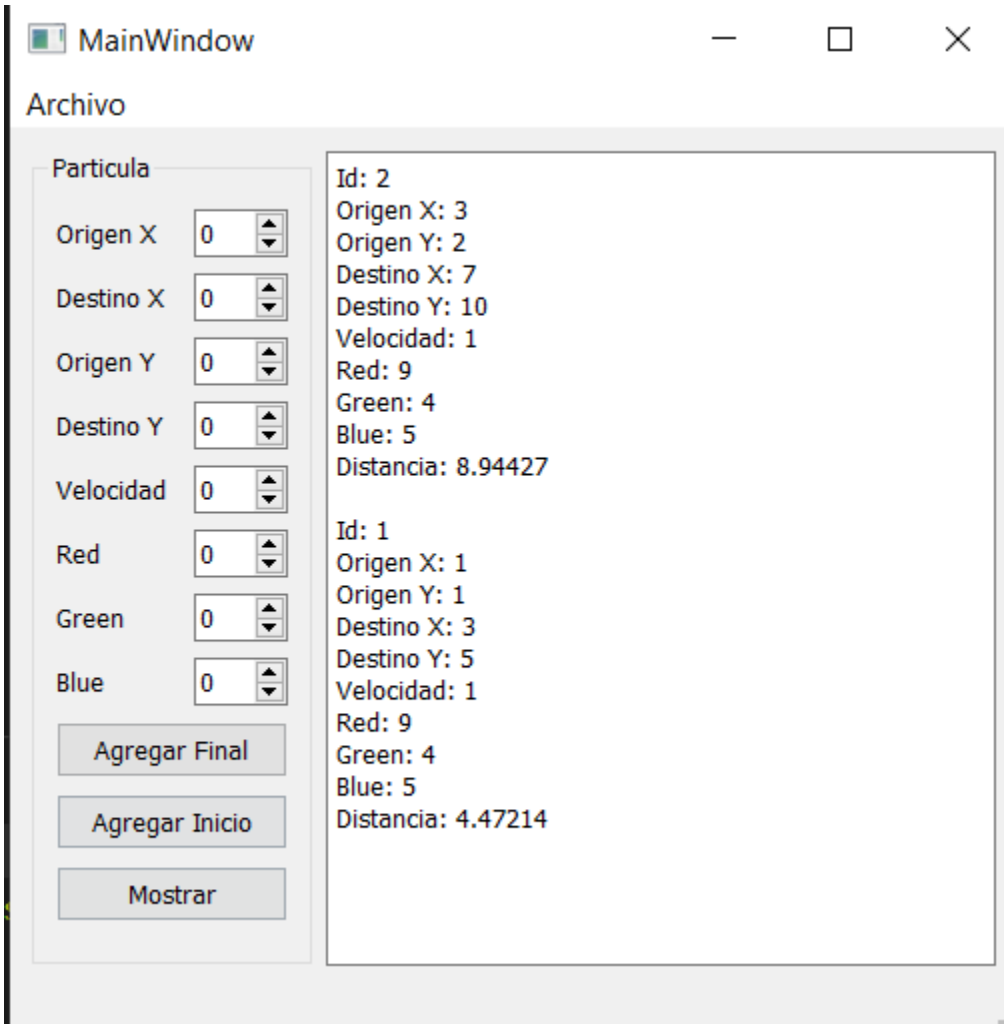

Se muestran capturas de pantallas de los pasos que se realizan en la interfaz para abrir el archivo de respaldo *.json*.







Muestra la captura de pantalla de las partículas con el método mostrar() después de abrir el respaldo



Conclusiones

Una actividad muy interesante en su totalidad, me gustó el uso de una interfaz y como poco a poco va creciendo el nivel de las cosas que estamos haciendo, nos adentramos a un mundo de posibilidades una vez que sabemos usar las herramientas que nos da python para el uso de la programación orientada a objetos, espero que en un futuro podamos seguir trabajando con esto, creando cada vez más cosas teniendo como

límite únicamente nuestra imaginación.

Referencias

undefined [MICHEL DAVALOS BOITES]. (2020, October 7). PySide2 - Introducción (Qt for Python)(I) [Video]. YouTube. Retrieved September 29, 2022, from https://www.youtube.com/watch?v=T0qJdF1fMqo&t=424s

Código

mainwindow.py

```
import imp
import re
from PySide2.QtWidgets import QMainWindow, QFileDialog, QMessageBox
from PySide2.QtCore import Slot
from ui_mainwindow import Ui_MainWindow
from particulasact.particula import Particula
from particulasact.index import Nodo, Lista_ligada



class MainWindow(QMainWindow):
    def __init__(self):
        super(MainWindow, self).__init__()
        self.lista_ligada = Lista_ligada()
        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)
        self.ui.agregarFinal_pushButton.clicked.connect(
            self.click_agregarFinal)
        self.ui.agregarInicio_pushButton.clicked.connect(
            self.click_agregarInicio)
        self.ui.mostrar_pushButton.clicked.connect(self.click_mostrar)
        self.ui.actionAbrir.triggered.connect(self.action_abrir_archivo)

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


    def creadorDeParticulas(self):
        destinoX = self.ui.destinoX_spinBox.value()
```

```
        origenX = self.ui.origenX_spinBox.value()
        destinoY = self.ui.destinoY_spinBox.value()
        origenY = self.ui.origenY_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()
        return Particula(self.lista_ligada.no_elements+1, origenX, origenY,
                         destinoX, destinoY, velocidad, red, green, blue)

    @Slot()
    def action_abrir_archivo(self):
        ubicacion = QFileDialog.getOpenFileName(
            self,
            'Abrir Archivo',
            '.',
            'JSON (*.json)'
        )[0]
        if self.lista_ligada.abrir(ubicacion):
            QMessageBox.information(
                self,
                "Exito",
                "Se pudo crear el archivo" + ubicacion
            )
        else:
            QMessageBox.critical(self, "Error", "El archivo no pudo
crearse")
```

```
    @Slot()
    def action_guardar_archivo(self):
        ubicacion = QFileDialog.getSaveFileName(
            self,
            'Guardar Archivo',
            '.',
            'JSON (*.json)'
        )[0]
        if self.lista_ligada.guardar(ubicacion):
            QMessageBox.information(
                self, "Exito", "Se pudo crear el archivo"+ubicacion)
        else:
            QMessageBox.critical(
                self,
                "Error",
                "El archivo no se pudo crear"
            )


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


    @Slot()
    def click_agregarFinal(self):
        particula = self.creadorDeParticulas()
        nodo = Nodo(particula)
        self.lista_ligada.agregar_final(nodo)
```

```
        self.ui.salida.clear()
        self.ui.salida.insertPlainText("Agregado al Final")


    @Slot()
    def click_agregarInicio(self):
        particula = self.creadorDeParticulas()
        nodo = Nodo(particula)
        self.lista_ligada.agregar_inicio(nodo)
        self.ui.salida.clear()
        self.ui.salida.insertPlainText("Agregado al Inicio")
```

index.py

```
import json
from particulasact.particula import Particula



class Nodo():
    dato = None
    siguiente = None
    anterior = None

    def __init__(self, dato):
        self.dato = dato
        self.siguiente = None



class Lista_ligada():
```

```
    nodo_inicial = None
    nodo_final = None
    no_elements = 0

    def __init__(self):
        self.nodo_inicial = None
        self.nodo_final = None

    def agregar_inicio(self, nodo):
        if(self.no_elements == 0):
            self.nodo_inicial = nodo
            self.nodo_final = nodo
            self.no_elements = self.no_elements + 1
        else:
            temporal = self.nodo_inicial
            temporal.anterior = nodo
            nodo.siguiente = temporal
            self.nodo_inicial = nodo
            self.no_elements = self.no_elements + 1

    def agregar_final(self, nodo):
        if(self.no_elements == 0):
            self.nodo_inicial = nodo
            self.nodo_final = nodo
            self.no_elements = self.no_elements + 1
        else:
            temporal = self.nodo_final
            temporal.siguiente = nodo
```

```
            nodo.anterior = temporal
            self.nodo_final = nodo
            self.no_elements = self.no_elements+1

    def mostrar(self):
        temp = self.nodo_inicial
        while(temp):
            print(temp.dato)
            temp = temp.siguiente

    def __str__(self):
        temp = self.nodo_inicial
        array = []
        while(temp):
            array.append(str(temp.dato))
            temp = temp.siguiente
        return "".join(array)

    def guardar(self, ubicacion):
        temp = self.nodo_inicial
        try:
            with open(ubicacion, 'w') as archivo:
                lista = []
                while(temp):
                    lista.append(temp.dato.to_dict())
                    temp = temp.siguiente
                json.dump(lista, archivo, indent=1)
            return 1
```

```
        except:
            return 0


    def abrir(self, ubicacion):
        try:
            with open(ubicacion, 'r') as archivo:
                lista = json.load(archivo)
                for particula in lista:
                    particulaNodo = Particula(**particula)
                    nodo = Nodo(particulaNodo)
                    self.agregar_final(nodo)
            return 1


        except:
            return 0
```