Actividad 09 – (QScene)

Rubio Valenzuela Miguel Angel - 216567795

Seminario de Algoritmia – D02.

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.

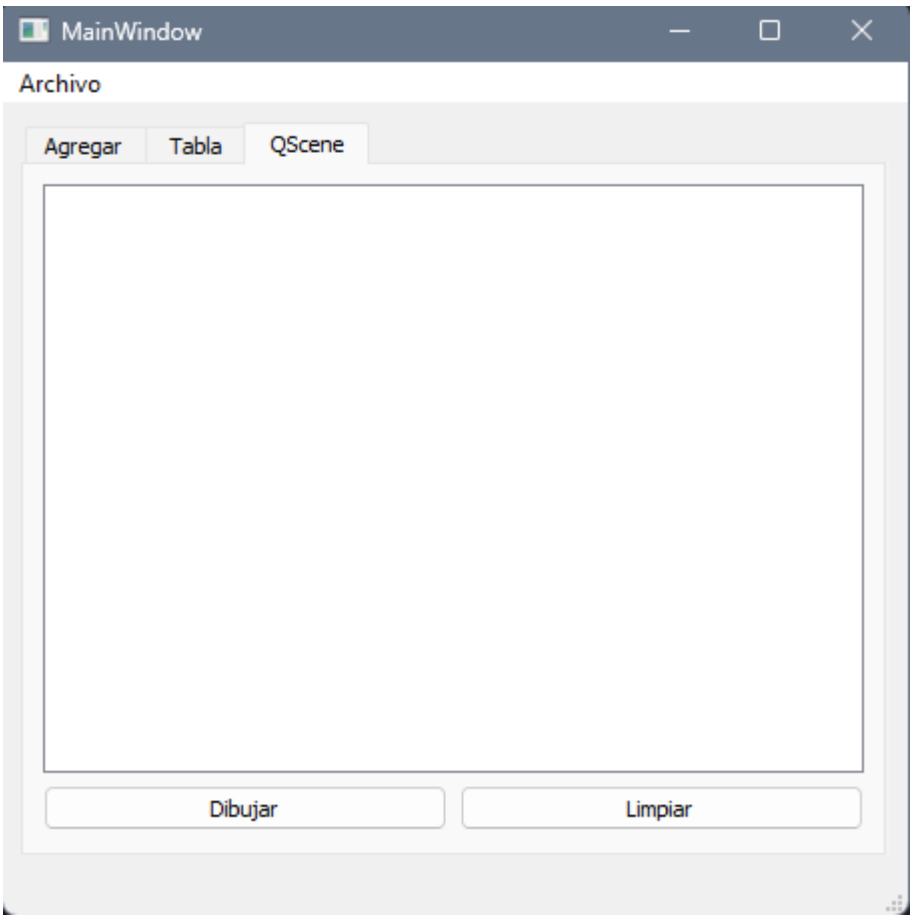
Desarrollo.

Para desarrollar el programa de esta actividad, se reutilizará el programa realizado en la actividad pasada, así como se realizó en las actividades anteriores.

Tenemos que crear una nueva pestaña en la interfaz que teníamos, le agregue el nombre de QScene, para que la pestaña funcionase, todo esto se realiza en el QtDesigner.

Después de hacer esto, nosotros tenemos que hacer que el diseño de la interfaz tenga un Graphics view y dos push button.

Estos cambios se ven reflejados de la siguiente manera.



Esta es la interfaz, y es la manera en la que quedaría completa

Pero el problema es que, no tienen funcionalidad, para esto tenemos que dirigirnos a la parte del código y darles por lo menos la opción de imprimir en consola el valor de limpiar y de dibujar.

```
PS C:\Users\cober\Desktop\Sem de Alg
sktop\Sem de Algoritmia\A9'; & 'C:\U
on\Python310\python.exe' 'c:\Users\c
on-2022.18.1\pythonFiles\lib\python\
' '65278' '--' 'c:\Users\cober\Deskt
py'
Dibujar
Limpiar
```

Así podemos confirmar que cuando presionamos un botón lo que envía a la consola es la información anterior, dibujar y limpiar.

Y con esto ya le hemos dado funcionalidad a los dos botones, y ahora tenemos que hacer que aparte de que muestren esa información lo que harán será dibujar una línea en la parte superior a los botones.

Lo que tenemos que hacer es crear la escena, para esto ingresamos el siguiente comando.

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

Aquí lo que realizamos es lo siguiente, creamos la escena con la primer línea y luego la ingresamos dentro de nuestra escena en la interfaz en la segunda línea.

Lo siguiente a realizar es hacer que al momento de ingresar a la función dibujar, este haga que se añadan líneas y círculos en la pantalla.

Usando las siguientes líneas de comando podemos mostrar una linea relacionada con dos puntos.

```python
@Slot()
def dibujar(self):
    pen = QPen()
    pen.setWidth(2)
    r = 100
    g = 200
    b = 50
    pen.setColor(QColor(r, g, b))

    self.scene.addEllipse(0+3,0+3,3,3, pen)
    self.scene.addEllipse(500,500,0,0, pen)
    self.scene.addLine(0+3,0+3,500,500, pen)
```

Primero, tenemos que usar la variable pen, como lápiz, además le agregamos un setWidth para controlar lo ancho del pincel

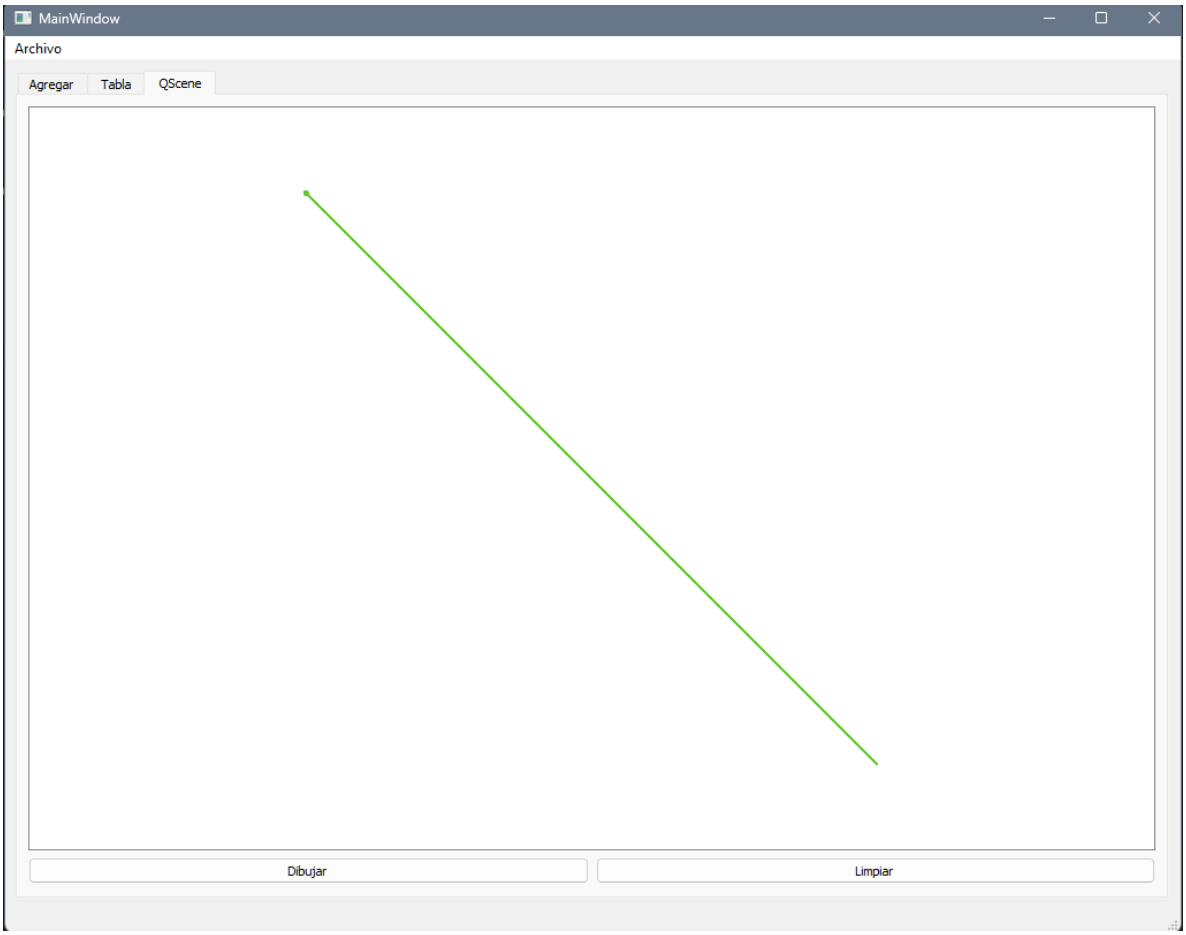
Usamos R G y B para tener los colores que va a tener el lápiz, y le añadimos la configuración a el lápiz,

Luego añadimos a la escena una elipse en la posición (3,3) y con un radio de 3 y le agregamos la configuración del lápiz.

Añadimos otra elipse en la posición (500, 500) y con un radio de 0, 0, con la configuración de pen

Ahora, usamos la función addLine que añade una línea en dos círculos desde la posición, (3,3) a la posición final (500, 500).

Y este es el resultado de implementar estas funciones.



Aquí se observa la línea marcada en el Scene.

Lo siguiente para limpiar la escena, tenemos que implementar la siguiente linea.

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

Y con esto, la escena anterior pasa a limpiarse, por lo que no veremos la linea otra vez, hasta que demos clic en el botón dibujar.

Ahora lo que tenemos que hacer es llamar a los valores que se encuentran en la partícula, para que estos valores se reflejen en la escena.

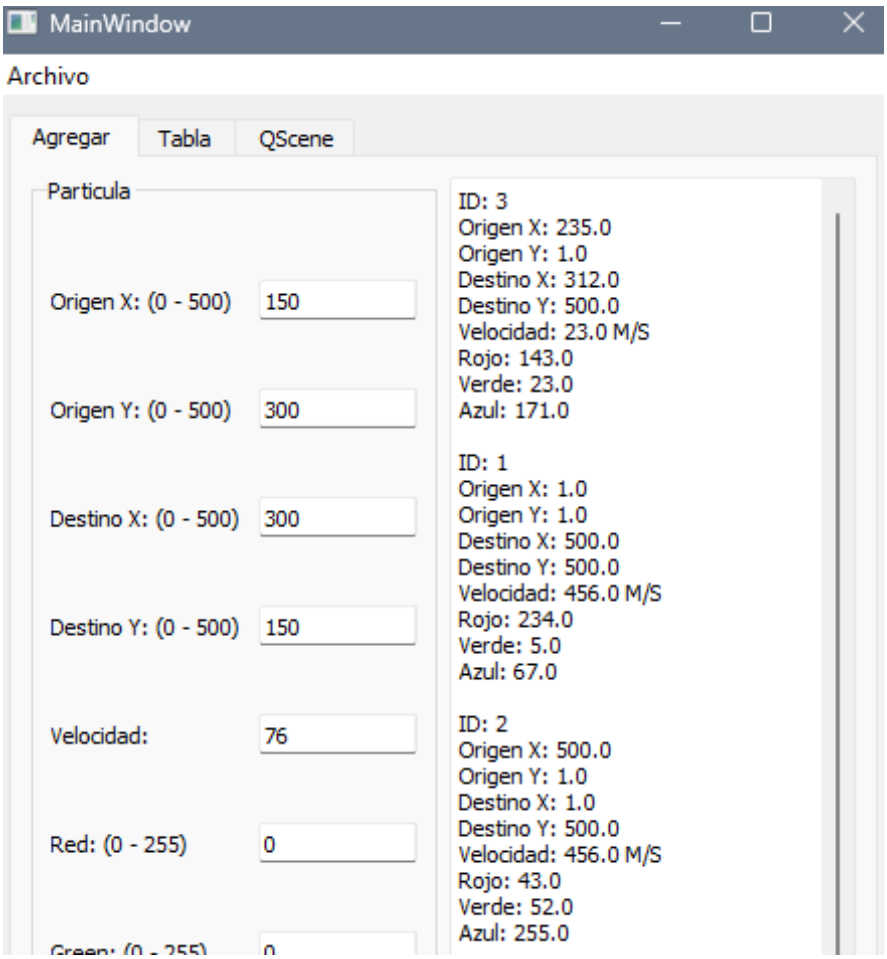
Para que esto funcione, tenemos que usar el objeto partícula e iterarlo para que se muestren estas partículas en la pantalla del scene.

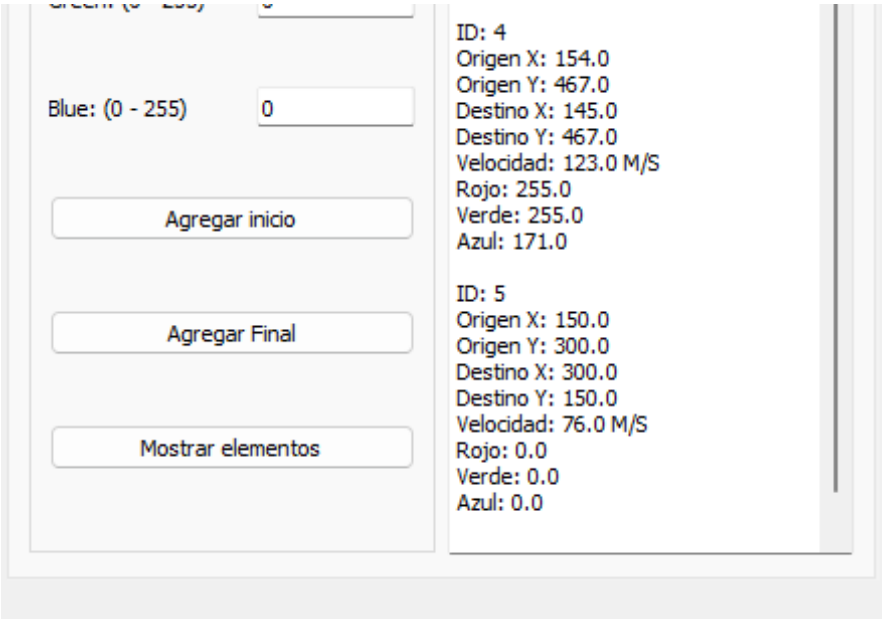
```
@Slot()
def dibujar(self):
    pen = QPen()
    pen.setWidth(4)

    for particula in self.lista:
        pen.setColor(QColor(float(particula.red),float(particula.green), float(particula.blue)))
        self.scene.addEllipse(float(particula.origenX),float(particula.origenY),3,3, pen)
        self.scene.addEllipse(float(particula.destinoX),float(particula.destinoY),3,3, pen)
        self.scene.addLine(float(particula.origenX),float(particula.origenY),float(particula.destinoX),float(particula.destinoY), pen)
```

De esta forma, iteramos en todas las partículas de nuestra lista y de esta forma nosotros podemos dibujarlas en la escena.

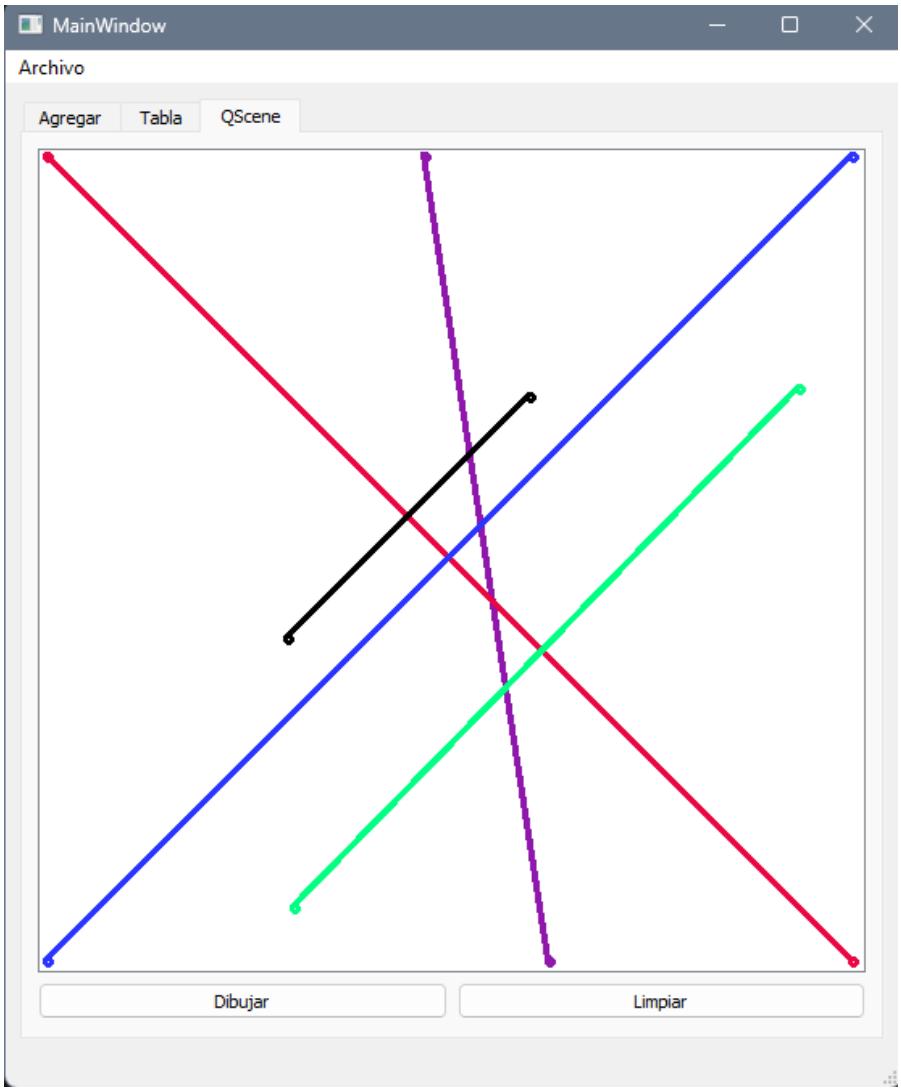
Para comprobar esto, voy a ingresar 5 distintos registros de particulas y apartir de ellos los voy a dibujar.

Primero vamos a ingresarlos.





Ahora, lo que debemos de hacer es una vez esos registros se encuentren ingresado, ir al apartado de QScene y dibujar las partículas y su recorrido.



Conclusiones

Al momento de desarrollar la actividad, el único inconveniente que yo genere es el hecho de que tuve que realizar la conversión de los valores mostrados como string en la pestaña de agregar, porque yo los quería convertir a enteros, pero el compilador no me dejaba poder realizar esa conversión, por lo que tuve que comprobar si iban a funcionar los diferentes métodos utilizando flotantes, y al momento de comprobar, me di cuenta de que si, el método QColor, addEllipse y addLine, también permiten que los valores, además de enteros sean flotantes.

De ahí en más, no tuve muchos inconvenientes al momento de desarrollar esta actividad, por lo que considero que estuvo algo mas sencilla que actividades pasadas.

Considero que aprender este tipo de métodos, nos ayudan a conocer de una manera práctica, como es que lo que nosotros desarrollamos puede convertirse en algo que yo considero como algo mas interactivo, porque ya conocemos como es la interacción que tienen los registros antes integrados a un programa que muestra la distancia y trayectoria que tiene.

Referencias

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

Código.

Al igual que en actividades pasadas, hay 5 archivos con códigos a mostrar

UserInterface.py

```
from PySide2.QtWidgets import QApplication
from mainwindow import MainWindow
import sys
#Se crea la aplicacion de QT
app = QApplication()

#Se crea un botton con la palabra hola
window = MainWindow()

#Se hace visible el boton.
window.show()

#QT Loop
sys.exit(app.exec_())
```

Particula.py

```python
from cmath import sqrt
import math

class Particula:
    def __init__(self, id = 0.0, origen_X= 0.0, origen_Y= 0.0,
                 destino_X= 0.0, destino_Y= 0.0, velocidad= 0.0,
                 red= 0.0, green= 0.0, blue= 0.0):
        self.__id = str(id)
        self.__origen_x = str(origen_X)
        self.__origen_y = str(origen_Y)
        self.__destino_x = str(destino_X)
        self.__destino_y = str(destino_Y)
        self.__velocidad = str(velocidad)
        self.__red = str(red)
        self.__green = str(green)
        self.__blue = str(blue)

    def __str__(self):
        return(
            'ID: ' + self.__id + '\n' +
            'Origen X: ' + self.__origen_x + '\n' +
            'Origen Y: ' + self.__origen_y + '\n' +
            'Destino X: ' + self.__destino_x + '\n' +
            'Destino Y: ' + self.__destino_y + '\n' +
            'Velocidad: ' + self.__velocidad + ' M/S \n' +
            'Rojo: ' + self.__red + '\n' +
            'Verde: ' + self.__green + '\n' +
            'Azul: ' + self.__blue + '\n'
            #'Distancia: ' + self.__distancia +  ' M \n'
            )

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

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

    @property
    def origenY(self):
        return self.__origen_y
```

```python
    @property
    def destinoX(self):
        return self.__destino_x

    @property
    def destinoY(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):
        ox = float(self.__origen_x)
        oy = float(self.__origen_y)
        dx = float(self.__destino_x)
        dy = float(self.__destino_y)
        return repr(sqrt(pow((ox - dx), 2) + pow((oy - dy), 2)))

    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
        }
```

MainWindow.py

```
from wsgiref import headers
from PySide2.QtWidgets import QMainWindow, QFileDialog, QMessageBox,
QTableWidgetItem, QGraphicsScene
from PySide2.QtCore import Slot
from ui_interfaz import Ui_MainWindow
from libreria import Lista
from PySide2.QtGui import QPen, QColor, QTransform
from particula import Particula
from random import randint

class MainWindow(QMainWindow):

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

        self.lista = Lista()
        self.id = int(0)
        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)
        self.ui.Agregar_inicio_pushButton.clicked.connect(self.click_agregar
_inicio)
        self.ui.Agregar_final_pushButton.clicked.connect(self.click_agregar_
final)
        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)

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

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

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

    @Slot()
    def dibujar(self):
        pen = QPen()
        pen.setWidth(4)

        for particula in self.lista:
```

```
            pen.setColor(QColor(float(particula.red),float(particula.green),
float(particula.blue)))
            self.scene.addEllipse(float(particula.origenX),float(particula.o
rigenY),3,3, pen)
            self.scene.addEllipse(float(particula.destinoX),float(particula.
destinoY),3,3, pen)
            self.scene.addLine(float(particula.origenX),float(particula.orig
enY),float(particula.destinoX),float(particula.destinoY), pen)

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

    @Slot()
    def mostrar_tabla(self):
        self.ui.tabla.setColumnCount(10)
        headers = ["ID","Orig X", "Orig Y", "Dest X", "Dest Y", "Velocidad"
,"Red",  "Green", "Blue", "Distancia"]
        self.ui.tabla.setHorizontalHeaderLabels(headers)

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

        row = 0
        for particula in self.lista:
            id_widget = QTableWidgetItem(particula.id)
            origenX_widget = QTableWidgetItem(particula.origenX)
            origenY_widget = QTableWidgetItem(particula.origenY)
            destinoX_widget = QTableWidgetItem(particula.destinoX)
            destinoY_widget = QTableWidgetItem(particula.destinoY)
            velocidad_widget = QTableWidgetItem(particula.velocidad)
            red_widget = QTableWidgetItem(particula.red)
            green_widget = QTableWidgetItem(particula.green)
            blue_widget = QTableWidgetItem(particula.blue)
            distancia_widget = QTableWidgetItem(particula.distancia)

            self.ui.tabla.setItem(row, 0, id_widget)
            self.ui.tabla.setItem(row, 1, origenX_widget)
            self.ui.tabla.setItem(row, 2, origenY_widget)
            self.ui.tabla.setItem(row, 3, destinoX_widget)
            self.ui.tabla.setItem(row, 4, destinoY_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 buscar_elemento(self):
        identificador = self.ui.buscar_lineEdit.text()

        find = False
        for particula in self.lista:
            if identificador == particula.id:
                self.ui.tabla.clear()
                self.ui.tabla.setColumnCount(10)
                headers = ["ID","Orig X", "Orig Y", "Dest X", "Dest Y",
                "Velocidad" ,"Red",  "Green", "Blue", "Distancia"]
                self.ui.tabla.setHorizontalHeaderLabels(headers)
                self.ui.tabla.setRowCount(1)

                id_widget = QTableWidgetItem(particula.id)
                origenX_widget = QTableWidgetItem(particula.origenX)
                origenY_widget = QTableWidgetItem(particula.origenY)
                destinoX_widget = QTableWidgetItem(particula.destinoX)
                destinoY_widget = QTableWidgetItem(particula.destinoY)
                velocidad_widget = QTableWidgetItem(particula.velocidad)
                red_widget = QTableWidgetItem(particula.red)
                green_widget = QTableWidgetItem(particula.green)
                blue_widget = QTableWidgetItem(particula.blue)
                distancia_widget = QTableWidgetItem(particula.distancia)

                self.ui.tabla.setItem(0, 0, id_widget)
                self.ui.tabla.setItem(0, 1, origenX_widget)
                self.ui.tabla.setItem(0, 2, origenY_widget)
                self.ui.tabla.setItem(0, 3, destinoX_widget)
                self.ui.tabla.setItem(0, 4, destinoY_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)

                find = True
                return

        if not find:
            QMessageBox.warning(
                self,
```

```
                "Atencion",
                "La particula" + identificador + "no encontrada"
            )



    @Slot()
    def action_abrir_archivo(self):
        dir = QFileDialog.getOpenFileName(
            self,
            "Abrir Archivo:",
            ".",
            "JSON (*.json)"
        )[0]
        if self.lista.abrir(dir):
            QMessageBox.information(
                self,
                "Exito",
                "Se abrió el archivo " + dir
            )
        else:
            QMessageBox.critical(
                self,
                "Error",
                "Error al abrir el archivo " + dir
            )

    @Slot()
    def action_guardar_archivo(self):
        #print("Guardando archivo")
        dir = QFileDialog.getSaveFileName(
            self,
            "Guardar como:",
            ".",
            "JSON (*.json)"
        )[0]

        if self.lista.guardar(dir):
            QMessageBox.information(
                self,
                "Éxito",
                "Se pudo crear y guardar datos del archivo " + dir
            )
        else:
            QMessageBox.critical(
```

```
                self,
                "Error",
                "No se pudo crear y/o guardar datos en el archivo " + dir
            )

    @Slot()
    def click_agregar_inicio(self):
        self.id = self.id + int(1)
        origenx = float(self.ui.Origen_X_lineEdit.text())
        origeny = float(self.ui.Origen_Y_lineEdit.text())
        destinox = float(self.ui.DestinoX_lineEdit.text())
        destinoy = float(self.ui.DestinoY_lineEdit.text())
        velocidad = float(self.ui.Velocidad_lineEdit.text())
        red = float(self.ui.Red_lineEdit.text())
        green = float(self.ui.Green_lineEdit.text())
        blue = float(self.ui.Blue_lineEdit.text())

        particula = Particula(self.id, origenx, origeny, destinox, destinoy,
velocidad, red, green, blue)

        self.lista.agregar_inicio(particula)

    @Slot()
    def click_agregar_final(self):
        self.id = self.id + int(1)
        origenx = float(self.ui.Origen_X_lineEdit.text())
        origeny = float(self.ui.Origen_Y_lineEdit.text())
        destinox = float(self.ui.DestinoX_lineEdit.text())
        destinoy = float(self.ui.DestinoY_lineEdit.text())
        velocidad = float(self.ui.Velocidad_lineEdit.text())
        red = float(self.ui.Red_lineEdit.text())
        green = float(self.ui.Green_lineEdit.text())
        blue = float(self.ui.Blue_lineEdit.text())

        particula = Particula(self.id, origenx, origeny, destinox, destinoy,
velocidad, red, green, blue)

        self.lista.agregar_final(particula)

        #self.ui.Salida.insertPlainText()

    @Slot()
    def click_mostrar(self):

        self.ui.Salida.clear()
```

```
        self.ui.Salida.insertPlainText(str(self.lista))
```

libreria.py

```
from particula import Particula
import json

class Lista:
    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 __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, direccion):
```

```
        try:
            with open(direccion, 'w') as archivo:
                lista = [ particula.to_dict() for particula in
self.__particulas ]
                json.dump(lista, archivo, indent=5)
            return 1
        except:
            return 0

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

ui_interfaz.py

```
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(452, 453)
        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_4 = QGridLayout(self.centralwidget)
        self.gridLayout_4.setObjectName(u"gridLayout_4")
        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.Green_lineEdit = QLineEdit(self.groupBox)
        self.Green_lineEdit.setObjectName(u"Green_lineEdit")

        self.gridLayout.addWidget(self.Green_lineEdit, 12, 3, 1, 2)

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

        self.gridLayout.addWidget(self.Origen_Y_lineEdit, 1, 3, 1, 2)

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

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

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

        self.gridLayout.addWidget(self.DestinoX_lineEdit, 2, 3, 1, 2)

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

        self.gridLayout.addWidget(self.Blue, 14, 1, 1, 1)

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

        self.gridLayout.addWidget(self.Red_lineEdit, 8, 3, 1, 2)

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

        self.gridLayout.addWidget(self.Green, 12, 1, 1, 1)

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

        self.gridLayout.addWidget(self.Velocidad_lineEdit, 4, 3, 1, 2)
```

```
        self.DestinoY_lineEdit = QLineEdit(self.groupBox)
        self.DestinoY_lineEdit.setObjectName(u"DestinoY_lineEdit")

        self.gridLayout.addWidget(self.DestinoY_lineEdit, 3, 3, 1, 2)

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

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

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

        self.gridLayout.addWidget(self.Origen_X_lineEdit, 0, 3, 1, 2)

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

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

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

        self.gridLayout.addWidget(self.Red, 8, 1, 1, 1)

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

        self.gridLayout.addWidget(self.DestinoY, 3, 1, 1, 2)

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

        self.gridLayout.addWidget(self.Blue_lineEdit, 14, 3, 1, 2)

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

        self.gridLayout.addWidget(self.Origen_X, 0, 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, 16, 1, 1,
4)

        self.Agregar_final_pushButton = QPushButton(self.groupBox)
        self.Agregar_final_pushButton.setObjectName(u"Agregar_final_pushButt
on")

        self.gridLayout.addWidget(self.Agregar_final_pushButton, 17, 1, 1,
4)

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

        self.gridLayout.addWidget(self.Mostrar_pushButton, 18, 1, 1, 4)


        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_3 = QGridLayout(self.tab_2)
        self.gridLayout_3.setObjectName(u"gridLayout_3")
        self.tabla = QTableWidget(self.tab_2)
        self.tabla.setObjectName(u"tabla")

        self.gridLayout_3.addWidget(self.tabla, 0, 0, 1, 3)

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

        self.gridLayout_3.addWidget(self.mostrar_tabla_pushButton, 1, 2, 1,
1)

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

        self.gridLayout_3.addWidget(self.buscar_pushButton, 1, 1, 1, 1)
```

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

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

        self.tabWidget.addTab(self.tab_2, "")
        self.tab_5 = QWidget()
        self.tab_5.setObjectName(u"tab_5")
        self.gridLayout_5 = QGridLayout(self.tab_5)
        self.gridLayout_5.setObjectName(u"gridLayout_5")
        self.graphicsView = QGraphicsView(self.tab_5)
        self.graphicsView.setObjectName(u"graphicsView")

        self.gridLayout_5.addWidget(self.graphicsView, 0, 0, 1, 2)

        self.dibujar_pushButton = QPushButton(self.tab_5)
        self.dibujar_pushButton.setObjectName(u"dibujar_pushButton")

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

        self.limpiar_pushButton = QPushButton(self.tab_5)
        self.limpiar_pushButton.setObjectName(u"limpiar_pushButton")

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

        self.tabWidget.addTab(self.tab_5, "")

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

        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QMenuBar(MainWindow)
        self.menubar.setObjectName(u"menubar")
        self.menubar.setGeometry(QRect(0, 0, 452, 22))
        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+O", 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(QCoreApplication.translate("MainWindow",
u"Particula", None))
        self.DestinoX.setText(QCoreApplication.translate("MainWindow",
u"Destino X: (0 - 500) ", None))
        self.Blue.setText(QCoreApplication.translate("MainWindow", u"Blue:
(0 - 255)", None))
        self.Green.setText(QCoreApplication.translate("MainWindow", u"Green:
(0 - 255)", None))
        self.Velocidad.setText(QCoreApplication.translate("MainWindow",
u"Velocidad:", None))
        self.Origen_Y.setText(QCoreApplication.translate("MainWindow",
u"Origen Y: (0 - 500)", None))
        self.Red.setText(QCoreApplication.translate("MainWindow", u"Red: (0
- 255)", None))
        self.DestinoY.setText(QCoreApplication.translate("MainWindow",
u"Destino Y: (0 - 500) ", None))
        self.Origen_X.setText(QCoreApplication.translate("MainWindow",
u"Origen X: (0 - 500)", None))
        self.Agregar_inicio_pushButton.setText(QCoreApplication.translate("M
ainWindow", u"Agregar inicio", None))
        self.Agregar_final_pushButton.setText(QCoreApplication.translate("Ma
inWindow", u"Agregar Final", None))
```

```
        self.Mostrar_pushButton.setText(QCoreApplication.translate("MainWind
ow", u"Mostrar elementos", None))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab),
QCoreApplication.translate("MainWindow", u"Agregar", None))
        self.mostrar_tabla_pushButton.setText(QCoreApplication.translate("Ma
inWindow", u"Mostrar", None))
        self.buscar_pushButton.setText(QCoreApplication.translate("MainWindo
w", u"Buscar", None))
        self.buscar_lineEdit.setPlaceholderText(QCoreApplication.translate("
MainWindow", u"ID de la particula.", None))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab_2),
QCoreApplication.translate("MainWindow", u"Tabla", None))
        self.dibujar_pushButton.setText(QCoreApplication.translate("MainWind
ow", u"Dibujar", None))
        self.limpiar_pushButton.setText(QCoreApplication.translate("MainWind
ow", u"Limpiar", None))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab_5),
QCoreApplication.translate("MainWindow", u"QScene", None))
        self.menuArchivo.setTitle(QCoreApplication.translate("MainWindow",
u"Archivo", None))
    # retranslateUi
```