Actividad 06 – (QPlainTextEdit)

Rubio Valenzuela Miguel Angel - 216567795

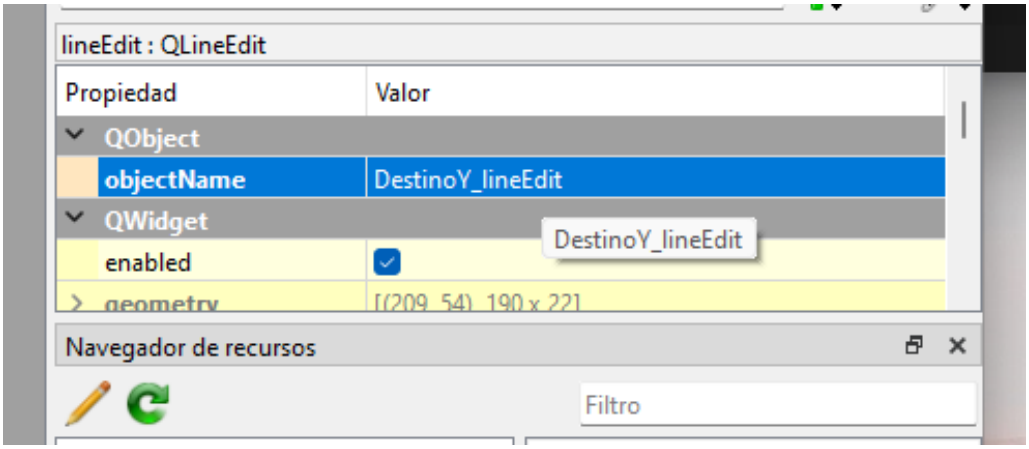
Seminario de Algoritmia – D02.

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 la captura de pantalla de los datos antes de usar el botón para `agregar_inicio()` y la captura de pantalla del mostrar partículas en el `QPlainTextEdit` después de haber agregado la `Particula`.
- [ ] Se muestra la captura de pantalla de los datos antes de usar el botón para `agregar_final()` y la captura de pantalla del mostrar partículas en el `QPlainTextEdit` después de haber agregado la `Particula`.

Desarrollo.

Lo primero para realizar es el cambio de nombre de valores de la interfaz esto para poder reconocer mejor todas las líneas de texto.

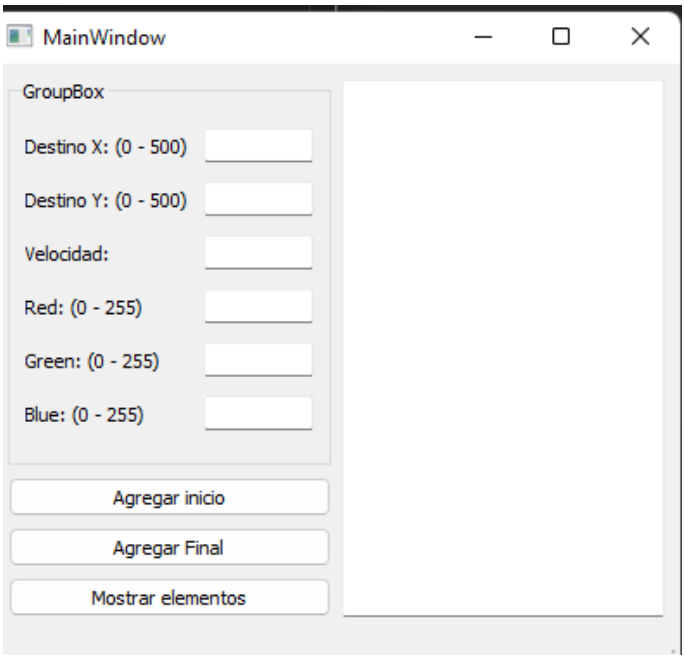


Ahí podemos observar el cambio que se ha ejecutado al nombre, todo esto lo realicé siguiendo las indicaciones del video del profe.

Esto se hace para todas las partes de la interfaz, teniendo así un orden al momento de controlar estas variables.

Después de realizar el cambio de nombres, lo que tuve que realizar es la generación del archivo ui_interfaz.py, a través del archivo interfaz.ui y esto se genera con el comando pyside2-uic interfaz.ui .o ui_interfaz.py.

Y después de esto, la interfaz quedaría de la siguiente manera.



Así es como termina la interfaz que desarrollé.

Después de eso, le agregue funcionalidad a los botones, de igual forma lo único que hice, de momento fue que hiciera la acción de mostrar click en la terminal.

Ahora lo que hice fue asignar los valores de la interfaz a variables internas dentro del programa, pero para poder realizar operaciones lo que hice fue así como se nos marca en el video, pero lo convertí a float, para que de esta forma se pudiese usar los valores en suma y realizar la operación de distancia euclidiana.

Ahora lo que sigue es nombrar de una manera el PlainTextEdit en este caso lo hice como en el video poniéndole como nombre salida.

Importar archivos con clases particula y librería.

Después de hacer esto tenemos que realizar la importación de las clases librería y particula que esta es la forma en las cuales lo llame.

Para esto realice: from librería import Lista, que es la clase que tenemos en esta librería.

```
from libreria import Lista
from particula import Particula
```

Y con esto, se puede usar la información de estos archivos, en nuestro archivo mainwindow, para así poder crear el objeto particula y poder ingresar la información ahí.

```
particula = Particula(id, 0.0, 0.0, destinox, destinoy, velocidad, red, green, blue)
```

Ademas, vamos a llamar a la clase Lista, esto se hace desde el constructor de la clase MainWindow para que la lista se genere una única vez y se le puedan agregar datos a esta lista.

```
Self.lista = Lista()
```

Y esto sirve para mandar a llamar funciones que agregaran información anterior de tipo partícula.

```
self.lista.agregar_inicio(particula)
```

Ahora lo que se hace es exactamente lo mismo pero para la siguiente funcion que es la de agregar al final, y pues prácticamente es lo mismo que se tiene que realizar.

Ahora se muestra como es que quedó la función agregar al final.

```
@Slot()
def click_agregar_final(self):
    self.id = self.id + 1
    destinox = float(self.ui.DestinoX_lineEdit.text())
    destinoy = float(self.ui.DestinoY_lineEdit.text())
    velocidad = float(self
    red = float(self.ui.Re
    green = float(self.ui.Green_lineEdit.text())
    blue = float(self.ui.Blue_lineEdit.text())

    particula = Particula(self.id, 0.0, 0.0, destinox, destinoy, velocidad, red, green, blue)

    self.lista.agregar_final(particula)
```

(variable) Green_lineEdit: QLineEdit

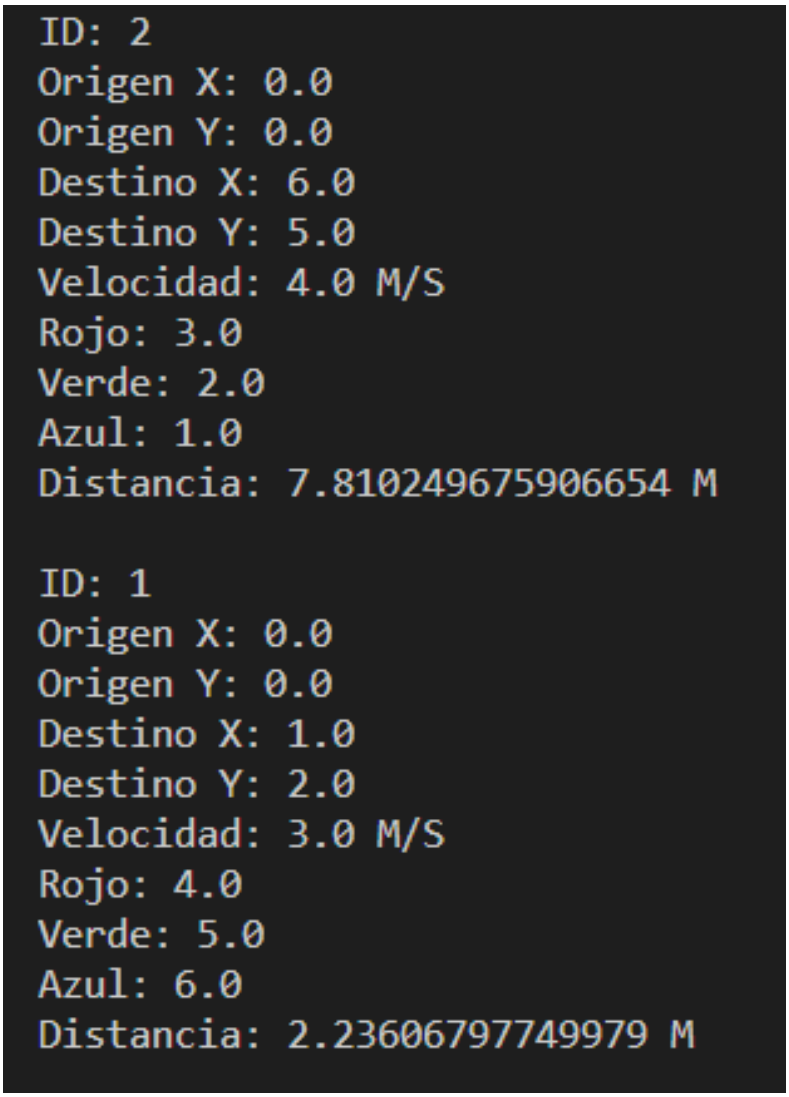
Esta es la manera en la que codifique esta parte, convirtiendo los valores en flotantes para poder realizar la operación de distancia.

De igual manera, ingrese dos datos utilizando la función de mostrar, sin usar el plaintextedit y esta es la forma en la que se puede observar el resultado en la consola, además de mostrar como se ve nuestro código.

```
@Slot()
def click_mostrar(self):

    self.lista.mostrar()
```

Y este es el resultado de como se muestra en la consola de visual studio.



Esto ocurre cuando el primer dato lo inserte al final, y el segundo dato lo inserte al inicio y mostré los resultados en la consola.

Ahora lo que falta es hacer que los datos se muestren en el plaintext

Para esto se ingresa a la clase librería y usamos el mismo método que se uso en la clase particula,

def __str__(self):

y retornamos un código, algo complejo que nos ayuda a mostrar todo dentro de nuestra lista

Este código terminaría de la siguiente forma:

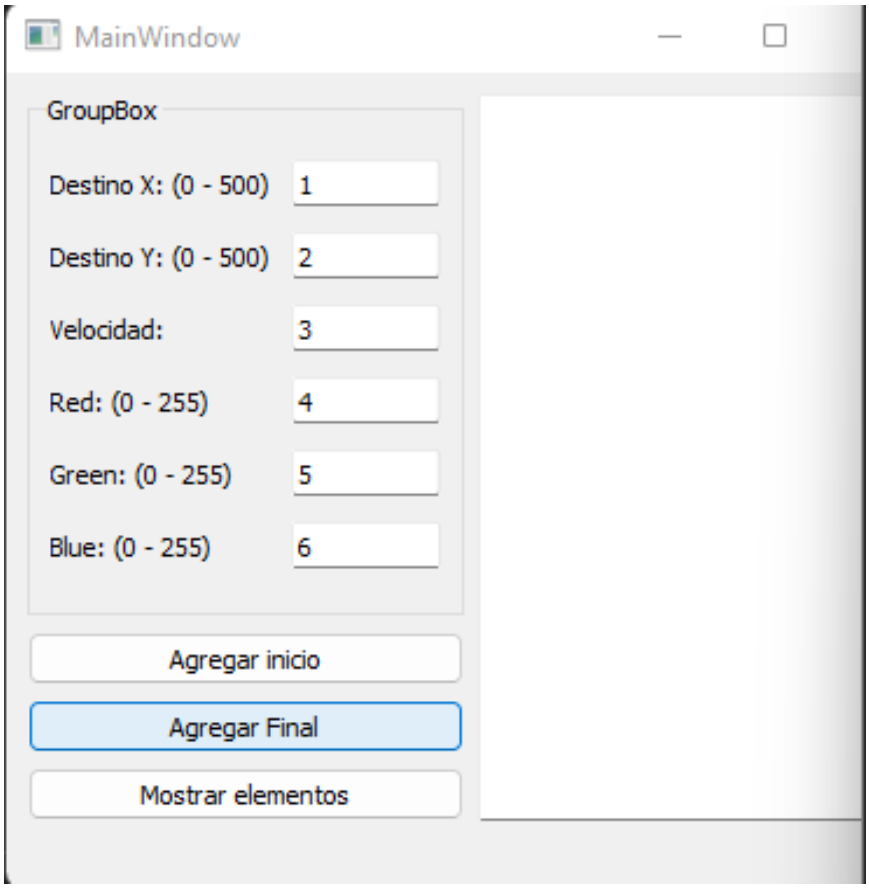
```
def __str__(self):
    return "".join(
        str(particula) for particula in self.__particulas
    )
```

Y nada mas realizamos la limpieza del plaintextedit para mostrar la información de manera mas limpia en la pantalla.

Asi de esta forma, terminamos de realizar el programa.

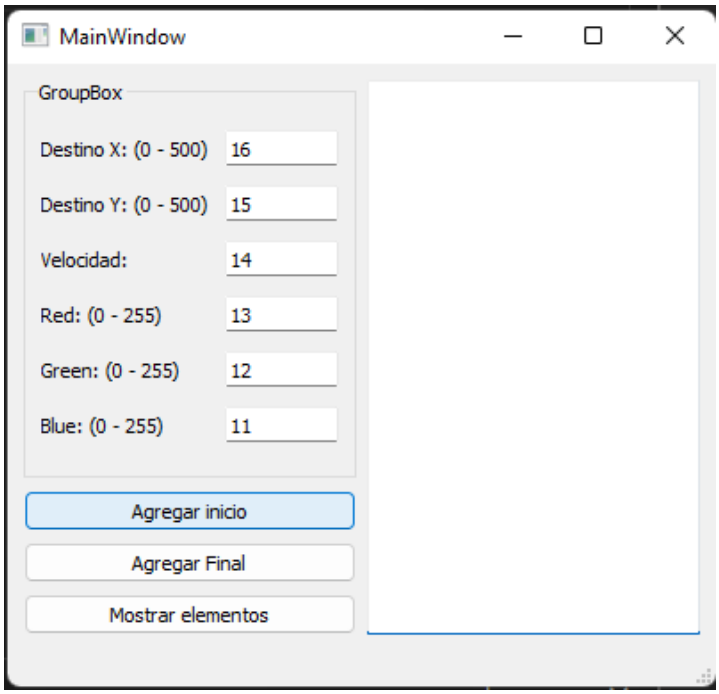
Lo único que falta es mostrar como es su funcionamiento, y todo el código completo se encontrará en la sección de código.

Estos son los resultados.

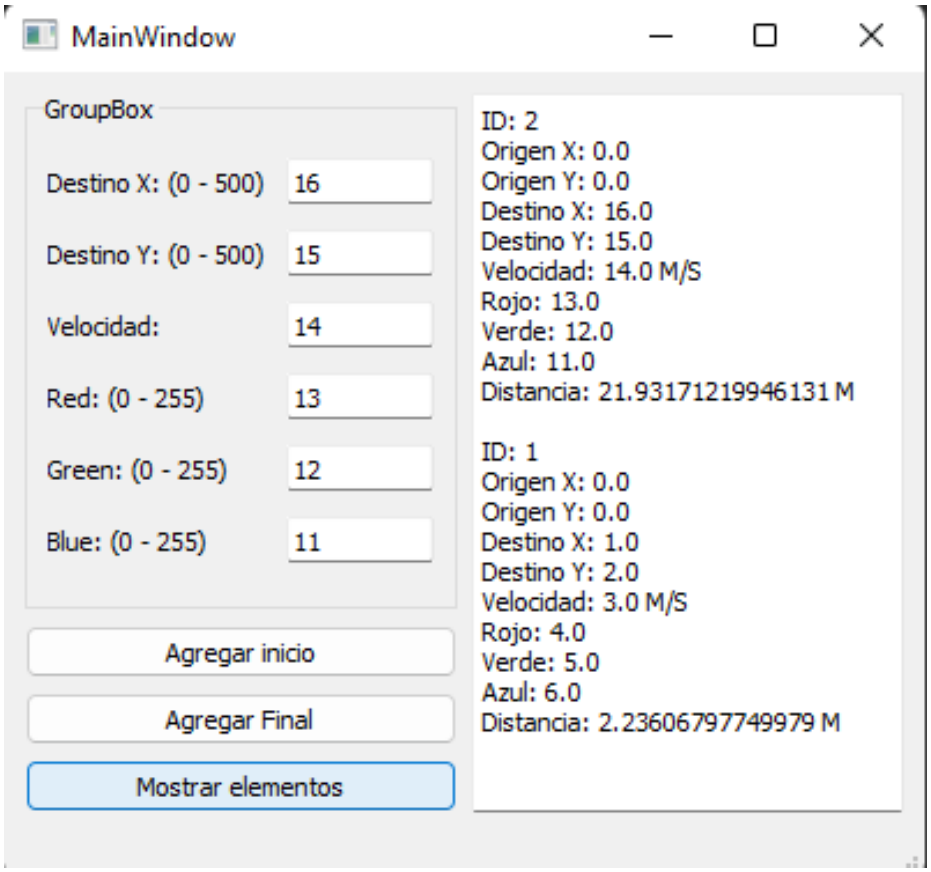


Agregamos la información con el botón, agregar al inicio.

Ahora agregamos otra partícula, pero esta va a ser agregada al inicio.



Y ya por último, mostramos la lista completa con los datos que ingresamos, y se mostraran en el orden agregado



Estos fueron los resultados de esta actividad y con esto terminamos.

Conclusiones

A decir verdad, no tuve mucho problema con la resolución de esta actividad, ya que teníamos que darle funcionamiento a los valores y a los botones que se encontraban en la interfaz que nosotros habíamos desarrollado.

Todos los problemas que presenté los resolví en base al video que nos proporcionó el profesor, y por lo tanto no presento ningún problema.

Creo que si se me complicó un poco el entender como es que vinculaban la interfaz con la información en otros archivos, pero al momento de realizar el cambio de nombre a las secciones en la interfaz supe como es que estas interactuaban, por lo que después de ahí se me hizo mas sencillo saber como es que podría seguir con el programa.

Una parte que no llegue a entender muy bien es la de la función __str__ dentro de la clase Lista, porque se me hizo raro que pusiera lo que se iba a mostrar en la parte de detrás del for, pero pues al ver que usa el join creo que lo que hace es concatenar la parte de detrás con la parte de adelante en ciclos.

Lo que nosotros podemos ver en una interfaz y lo que ocurre en la parte trasera de esta, son cosas completamente diferentes, y esto me esta dando una idea mucho más completa de como es que trabajan los programas y como es que podemos mostrar algo en la pantalla mientras que otros algoritmos pueden trabajar sin que uno pueda percatarse se esto.

Referencias

https://www.youtube.com/watch?v=5TPKrKlAAU0&t=21s&ab_channel=MICHELDAVALOSBOITES – Michel Davalos Boites – PySide2 - QPlainTextEdit (Qt for Python)(III)

Código.

Para este apartado, voy a agregar los 6 archivos que componen todo el programa de la interfaz.

1. **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_())
```

2. **Librería.py**

```
from particula import Particula

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
        )
```

3. 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(398, 349)
        self.centralwidget = QWidget(MainWindow)
        self.centralwidget.setObjectName(u"centralwidget")
        self.gridLayout_2 = QGridLayout(self.centralwidget)
        self.gridLayout_2.setObjectName(u"gridLayout_2")
        self.Mostrar_pushButton = QPushButton(self.centralwidget)
        self.Mostrar_pushButton.setObjectName(u"Mostrar_pushButton")

        self.gridLayout_2.addWidget(self.Mostrar_pushButton, 4, 0, 1, 1)

        self.Agregar_inicio_pushButton = QPushButton(self.centralwidget)
        self.Agregar_inicio_pushButton.setObjectName(u"Agregar_inicio_pushBu
tton")

        self.gridLayout_2.addWidget(self.Agregar_inicio_pushButton, 2, 0, 1,
1)

        self.groupBox = QGroupBox(self.centralwidget)
        self.groupBox.setObjectName(u"groupBox")
        self.gridLayout = QGridLayout(self.groupBox)
        self.gridLayout.setObjectName(u"gridLayout")
        self.Red_lineEdit = QLineEdit(self.groupBox)
        self.Red_lineEdit.setObjectName(u"Red_lineEdit")

        self.gridLayout.addWidget(self.Red_lineEdit, 6, 3, 1, 2)

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

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

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

```
        self.gridLayout.addWidget(self.Blue, 12, 1, 1, 1)

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

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

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

        self.gridLayout.addWidget(self.Green, 10, 1, 1, 1)

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

        self.gridLayout.addWidget(self.DestinoX, 0, 1, 1, 2)

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

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

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

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

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

        self.gridLayout.addWidget(self.Red, 6, 1, 1, 1)

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

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

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

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

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

```
        self.gridLayout.addWidget(self.Green_lineEdit, 10, 3, 1, 2)


        self.gridLayout_2.addWidget(self.groupBox, 1, 0, 1, 1)

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

        self.gridLayout_2.addWidget(self.Agregar_final_pushButton, 3, 0, 1,
1)

        self.Salida = QPlainTextEdit(self.centralwidget)
        self.Salida.setObjectName(u"Salida")

        self.gridLayout_2.addWidget(self.Salida, 1, 1, 4, 1)

        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QMenuBar(MainWindow)
        self.menubar.setObjectName(u"menubar")
        self.menubar.setGeometry(QRect(0, 0, 398, 22))
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QStatusBar(MainWindow)
        self.statusbar.setObjectName(u"statusbar")
        MainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(MainWindow)

        QMetaObject.connectSlotsByName(MainWindow)
    # setupUi

    def retranslateUi(self, MainWindow):
        MainWindow.setWindowTitle(QCoreApplication.translate("MainWindow",
u"MainWindow", None))
        self.Mostrar_pushButton.setText(QCoreApplication.translate("MainWind
ow", u"Mostrar elementos", None))
        self.Agregar_inicio_pushButton.setText(QCoreApplication.translate("M
ainWindow", u"Agregar inicio", None))
        self.groupBox.setTitle(QCoreApplication.translate("MainWindow",
u"GroupBox", None))
        self.DestinoY.setText(QCoreApplication.translate("MainWindow",
u"Destino Y: (0 - 500) ", None))
        self.Blue.setText(QCoreApplication.translate("MainWindow", u"Blue:
(0 - 255)", None))
```

```
        self.Velocidad.setText(QCoreApplication.translate("MainWindow",
u"Velocidad:", None))
        self.Green.setText(QCoreApplication.translate("MainWindow", u"Green:
(0 - 255)", None))
        self.DestinoX.setText(QCoreApplication.translate("MainWindow",
u"Destino X: (0 - 500) ", None))
        self.Red.setText(QCoreApplication.translate("MainWindow", u"Red: (0
- 255)", None))
        self.Agregar_final_pushButton.setText(QCoreApplication.translate("Ma
inWindow", u"Agregar Final", None))
    # retranslateUi
```

4. Mainwindow.py

```
from PySide2.QtWidgets import QMainWindow
from PySide2.QtCore import Slot
from algoritmos import distancia_euclidiana
from ui_interfaz import Ui_MainWindow
from libreria import Lista
from particula import Particula

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)

    @Slot()
    def click_agregar_inicio(self):
        self.id = self.id + int(1)
        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, 0.0, 0.0, destinox, destinoy,
velocidad, red, green, blue)

        self.lista.agregar_inicio(particula)

    @Slot()
    def click_agregar_final(self):
        self.id = self.id + int(1)
        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, 0.0, 0.0, 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))
```

5. Particula.py

```
from algoritmos import distancia_euclidiana


class Particula:
    def __init__(self, id=0.0, ox=0.0, oy=0.0, dx=0.0, dy=0.0, vel=0.0, r=0,
g=0, b=0):
        self.__id = repr(id)
        self.__origen_x = repr(ox)
        self.__origen_y = repr(oy)
        self.__destino_x = repr(dx)
```

```
        self.__destino_y = repr(dy)
        self.__velocidad = repr(vel)
        self.__red = repr(r)
        self.__green = repr(g)
        self.__blue = repr(b)
        self.__distancia = repr(distancia_euclidiana(ox, oy, dx, dy))

    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'
            )
```

6. Algoritmos.py

```
import math

def distancia_euclidiana(x_1, y_1, x_2, y_2):
    return math.sqrt( pow( x_1 - x_2, 2 ) + pow( y_1 - y_2, 2 ))
```