Actividad 11 – Fuerza bruta

Hernández Lomelí Diego Armando

Seminario de algoritmia 2022B D02

Lineamientos de evaluación.

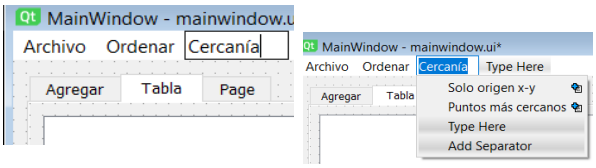
- [ ] El reporte está en formato Google Docs o PDF.
- [ ] El reporte sigue las pautas del Formato de Actividades .
- [ ] El reporte tiene desarrollada todas las pautas del Formato de Actividades.
- [ ] Se muestra captura de pantalla de los puntos de las partículas en el `QScene`.
- [ ] Se muestra captura de pantalla del resultado del algoritmo de fuerza bruta en el `QScene`.

Desarrollo

Para esta actividad tendremos que implementar un algoritmo de fuerza bruta para encontrar los puntos de origen más cercanos entre las partículas almacenadas.

Para ello vamos a agregar 2 acciones que nos permitan utilizar las funciones que necesitamos para nuestro cometido, la primera nos servirá para dibujar únicamente los puntos de origen, sin su destino, la otra acción nos permitirá dibujar las líneas entre los puntos que estén más cercanos.

Primeramente, agregamos las acciones adicionales a la interfaz que trabajamos en la clase anterior. Como nota adicional vamos a utilizar el componente "**QScene"** de la actividad 9 para dibujar los puntos de origen.



Ya con las acciones agregadas vamos a introducirnos a programar su funcionalidad no sin antes aclarar que para esta actividad usaremos los datos de prueba del archivo **particulas 2.json** en el siguiente enlace https://boites.notion.site/Actividad-11-Fuerza-Bruta-5e8f8972384c472aa8f86c42cbce3fd2, contiene 50 registros de prueba para testear los resultados.

Empezaremos con los puntos de origen de las partículas, su implementación es relativamente sencilla, pues ya la tenemos de la misma **actividad 9** la diferencia es que dibujaremos considerablemente menos elementos solo con esta función

```python
@Slot()
def action_solo_origenes(self):
    self.limpiar()
    """ Dibujamos todas las particulas a la vez """
    for particula in self.__lista:
        pen = QPen()
        pen.setWidth(2)

        color = QColor(particula.red, particula.green, particula.blue)
        pen.setColor(color)

        self.scene.addEllipse(float(particula.origen_x),
                              float(particula.origen_y), 3, 3, pen)
```

Si vemos bien, solo pintamos los puntos de origen y removemos los demás dibujos.

Lo siguiente es el método de fuerza bruta para encontrar los puntos más cercanos. El método será implementado dentro de la clase **listaParticula.**

Este código al ser de fuerza bruta ejecutará 2 bucles para encontrar la distancia mínima por cada elemento del arreglo, además, no tomará ninguna consideración para hacer más optimo o eficiente el trabajo, solo se preocupa por encontrar el resultado esperado sin importar la memoria disponible o el tiempo de ejecución.

Como contemplación adicional agregamos una variable identificada como **mínimo** para poder forzar que la primera comparación siempre sea la distancia más cercana

```
def puntos_cercanos(self):
    """ ALmacenará los resultados de la detección """
    resultado = []
    for punto_i in self.__particulas:
        x = punto_i.origen_x
        y = punto_i.origen_y
        """ utilizamos una distancia considerablemente grande para forzar que el punto
         de comparación siempre tenga una distancia inferior al inicio, despues se obtendrá
         el punto verdaderamente más cercano"""
        minimo = 1000
        cercano = (0, 0)
        color = (punto_i.red,
                 punto_i.green,
                 punto_i.blue)
        """ Segundo recorrido para hacer la comparación con el elemento actual """
        for punto_j in self.__particulas:
            """ Evitamos comparar el punto inicial con el mismo """
            if (punto_i != punto_j):
                """ Guardamos los nuevos puntos de comparación """
                x_comparison = punto_j.origen_x
                y_comparison = punto_j.origen_y
                """ Obtenemos la distancia euclidiana entre los 2 puntos para determinar si es
                    mayor o menor """
                d = distancia_euclidiana(x, y, x_comparison, y_comparison)
                if d < minimo:
                    """ Corregimos el valor de la distancia minima para proximas comparaciones """
                    minimo = d
                    """ Asignamos al punto más cercanos con los valores de comparación actuales """
                    cercano = (x_comparison, y_comparison)
        """ Agregamos al arreglo el punto inicial de comparación y depues el punto que se
        detectó como el más cercano """
        resultado.append(((x, y), cercano, color))
    """ Finalmente retornamos los resultados obtenidos """
    return resultado
```

Y almacenará en un arreglo los dos puntos de comparación y el color de la partícula de origen. Este ultimo atributo no es necesario, pero puede ser de gran ayuda para visualizar las conexiones entre puntos.

Ahora vamos a utilizar este método dentro de la clase **mainwindow.** Como el método fue trabajado dentro de **listaParticula** accederemos a él a través de la instancia de clase **__lista**.

```
@Slot()
def dibujar_mas_cercanos(self):
    puntos_cercanos = self.__lista.puntos_cercanos()
    for punto1, punto2, color in puntos_cercanos:
        pen = QPen()
        pen.setWidth(2)

        c = QColor(color[0], color[1], color[2])
        pen.setColor(c)
        self.scene.addLine(float(punto1[0]), float(punto1[1]),
                           float(punto2[0]), float(punto2[1]), pen)
```

Si notamos este método es muy similar al anterior, su diferencia es el origen de los datos, recibido en un arreglo obtenido a partir de la comparación de un punto de referencia con todos los demás puntos existentes en le lista de partículas, para dibujar la línea que une los puntos cercanos.

Resultado de ejecución.

Conclusiones

Esta actividad fue de bastante utilidad para entender que se puede realizar una tarea aparentemente sencilla en una que sea sencilla de entender, pero también notar que aunque el método sea capaz de realizar su cometido puede no ser la mejor implementación, en el caso de métodos de fuerza bruta lo consideramos como una posibilidad viable solamente en ciertas ocasiones en las que no tengamos el total conocimiento de lo que buscamos, pues a diferencia de otros métodos, estos numeran todas las posibles soluciones.

Referencia

BOITES, M. D. (18 de Octubre de 2021). Obtenido de Youtube: https://www.youtube.com/watch?v=hPARP71VEH0

Código

Código archivo **Main.py**

```
from PySide2.QtWidgets import QApplication
from mainwindow import MainWindow
import sys
app = QApplication()
window = MainWindow()
window.show()
sys.exit(app.exec_())
```

Código archivo **mainwindow.py**

```
from ui_mainwindow import Ui_MainWindow, QFileDialog, QMessageBox,
QTableWidgetItem, QPen, QColor, QGraphicsScene
from PySide2.QtWidgets import QMainWindow
from PySide2.QtCore import Slot
from listaParticulas import listaParticula
from Particula import Particula


class MainWindow(QMainWindow):
    __contador = 0

    def __init__(self):
        super(MainWindow, self).__init__()
        self.__lista = listaParticula()
        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)
        self.ui.btnAgregarInicio.clicked.connect(self.click_agregar_inicio)
```

```
        self.ui.btnAgregarFinal.clicked.connect(self.click_agregar_final)
        self.ui.btnMostrar.clicked.connect(self.mostrar)
        """ Metodos para el menu de la ventana """
        self.ui.actionAbrir.triggered.connect(self.action_abrir_archivo)
        self.ui.actionGuardar.triggered.connect(self.action_guardar_archivo)
        """ Metodos para ordenar """
        self.ui.actionSortId.triggered.connect(self.sortById)
        self.ui.actionSortDistance.triggered.connect(self.sortByDistance)
        self.ui.actionSortSpeed.triggered.connect(self.sortBySpeed)
        """ Metodos para encontrar la cercanía en los puntos de origen """
        self.ui.actionOnlyOrigenXY.triggered.connect(self.action_solo_origen
es)
        self.ui.actionNearestPoints.triggered.connect(
            self.dibujar_mas_cercanos)
        """ Metodos para trabajar con la tabla """
        self.ui.mostrar_pushButton.clicked.connect(self.mostrar_tabla)
        self.ui.buscar_pushButton.clicked.connect(self.buscar_id_tabla)

        self.ui.dibujarPushButton.clicked.connect(self.dibujar)
        self.ui.limpiarPushBtn.clicked.connect(self.limpiar)

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

    @Slot()
    def dibujar(self):
        """ Dibujamos todas las particulas a la vez """
        for particula in self.__lista:
            pen = QPen()
            pen.setWidth(2)

            color = QColor(particula.red, particula.green, particula.blue)
            pen.setColor(color)

            self.scene.addEllipse(float(particula.origen_x),
                                  float(particula.origen_y), 3, 3, pen)
            self.scene.addEllipse(float(particula.destino_x),
                                  float(particula.destino_y), 3, 3, pen)
            self.scene.addLine(float(particula.origen_x),
float(particula.origen_y),
                               float(particula.destino_x),
float(particula.destino_y), pen)

    def wheelEvent(self, event):
        if event.delta() > 0:
```

```
            self.ui.particulasView.scale(1.2, 1.2)
            return
        self.ui.particulasView.scale(0.8, 0.8)

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

    @Slot()
    def mostrar_tabla(self):
        self.ui.tableParticulas.setColumnCount(10)
        headers = ["id", "origen_x", "origen_y", "destino_x",
                   "destino_y", "velocidad", "red", "green", "blue",
"distancia"]
        self.ui.tableParticulas.setHorizontalHeaderLabels(headers)
        self.ui.tableParticulas.setRowCount(len(self.__lista))
        """ Empezamos a rellenar la tabla """
        row = 0
        for particula in self.__lista:
            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.tableParticulas.setItem(row, 0, id_widget)
            self.ui.tableParticulas.setItem(row, 1, origen_x_widget)
            self.ui.tableParticulas.setItem(row, 2, origen_y_widget)
            self.ui.tableParticulas.setItem(row, 3, destino_x_widget)
            self.ui.tableParticulas.setItem(row, 4, destino_y_widget)
            self.ui.tableParticulas.setItem(row, 5, velocidad_widget)
            self.ui.tableParticulas.setItem(row, 6, red_widget)
            self.ui.tableParticulas.setItem(row, 7, green_widget)
            self.ui.tableParticulas.setItem(row, 8, blue_widget)
            self.ui.tableParticulas.setItem(row, 9, distancia_widget)

            row += 1

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

```
        idBusqueda = self.ui.searchEdit.text()

        for particula in self.__lista:
            if idBusqueda == str(particula.id):
                self.ui.tableParticulas.clear()
                self.ui.tableParticulas.setColumnCount(10)
                headers = ["id", "origen_x", "origen_y", "destino_x",
                           "destino_y", "velocidad", "red", "green", "blue",
"distancia"]
                self.ui.tableParticulas.setHorizontalHeaderLabels(headers)
                self.ui.tableParticulas.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.tableParticulas.setItem(0, 0, id_widget)
                self.ui.tableParticulas.setItem(0, 1, origen_x_widget)
                self.ui.tableParticulas.setItem(0, 2, origen_y_widget)
                self.ui.tableParticulas.setItem(0, 3, destino_x_widget)
                self.ui.tableParticulas.setItem(0, 4, destino_y_widget)
                self.ui.tableParticulas.setItem(0, 5, velocidad_widget)
                self.ui.tableParticulas.setItem(0, 6, red_widget)
                self.ui.tableParticulas.setItem(0, 7, green_widget)
                self.ui.tableParticulas.setItem(0, 8, blue_widget)
                self.ui.tableParticulas.setItem(0, 9, distancia_widget)

                return
        QMessageBox.warning(
            self,
            "Error",
            f'No se ha encontrado una particula con el id: "{idBusqueda}"'
        )
```

```python
    """ Generación de eventos para acciones del menu """
    @Slot()
    def action_guardar_archivo(self):
        ubicacion = QFileDialog.getSaveFileName(
            self,
            "Guardar archivo",
            ".",
            "JSON (*.json)"
        )[0]
        if self.__lista.guardar(ubicacion):
            QMessageBox.information(
                self,
                "Exito",
                ("Se pudo crear el archivo " + ubicacion)
            )
        else:
            QMessageBox.critical(
                self,
                "Error",
                ("No pudo crear el archivo " + ubicacion)
            )

    @Slot()
    def action_abrir_archivo(self):
        ubicacion = QFileDialog.getSaveFileName(
            self,
            "Guardar archivo",
            ".",
            "JSON (*.json)"
        )[0]
        if self.__lista.abrir(ubicacion):
            QMessageBox.information(
                self,
                "Exito",
                ("Se pudo abrir el archivo " + ubicacion)
            )
            self.ui.plainTextEdit.clear()
            self.ui.plainTextEdit.insertPlainText(str(self.__lista))
        else:
            QMessageBox.critical(
                self,
                "Error",
                ("No pudo abrir el archivo " + ubicacion)
            )
```

```
    """ Generación de metodos para la cercanía d4e los puntos """
    @Slot()
    def action_solo_origenes(self):
        self.limpiar()
        """ Dibujamos todas las particulas a la vez """
        for particula in self.__lista:
            pen = QPen()
            pen.setWidth(2)

            color = QColor(particula.red, particula.green, particula.blue)
            pen.setColor(color)

            self.scene.addEllipse(float(particula.origen_x),
                                  float(particula.origen_y), 3, 3, pen)

    @Slot()
    def dibujar_mas_cercanos(self):
        puntos_cercanos = self.__lista.puntos_cercanos()
        for punto1, punto2, color in puntos_cercanos:
            pen = QPen()
            pen.setWidth(2)

            c = QColor(color[0], color[1], color[2])
            pen.setColor(c)
            self.scene.addLine(float(punto1[0]), float(punto1[1]),
                               float(punto2[0]), float(punto2[1]), pen)

    @ Slot()
    def click_agregar_inicio(self):
        self.__lista.agregar_inicio(self.procesarParticula())
        self.__contador += 1

    @ Slot()
    def click_agregar_final(self):
        self.__lista.agregar_final(self.procesarParticula())
        self.__contador += 1

    @ Slot()
    def mostrar(self):
        self.ui.plainTextEdit.clear()
        self.ui.plainTextEdit.insertPlainText(str(self.__lista))

    @ Slot()
    def sortById(self):
        self.__lista.sortById()
```

```
    @ Slot()
    def sortByDistance(self):
        self.__lista.sortByDistance()

    @ Slot()
    def sortBySpeed(self):
        self.__lista.sortBySpeed()

    def procesarParticula(self):
        """ id, origen_x, origen_y, destino_x, destino_y, velocidad, red,
green, blue, distancia """
        return Particula(self.__contador,
                         self.ui.spnnOrigenX.value(),
                         self.ui.spnnOrigenY.value(),
                         self.ui.spnnDestinoX.value(),
                         self.ui.spnnDestinoY.value(),
                         int(self.ui.spnnVelocidad.text()),
                         self.ui.spnnRed.value(),
                         self.ui.spnnBlue.value(),
                         self.ui.spnnGreen.value(),
                         self.ui.spnnDistancia.value())
```

Código archivo **Particula.py**

```
from algoritmos import distancia_euclidiana


class Particula(object):
    __id = 0
    __origen_x = 0
    __origen_y = 0
    __destino_x = 0
    __destino_y = 0
    __velocidad = 0
    __red = 0
    __green = 0
    __blue = 0
    __distancia = 0.0

    def __init__(self, id, origen_x, origen_y, destino_x, destino_y,
velocidad, red, green, blue):
        """ Propiedades de la clase """
        self.__id = id
```

```
        self.__origen_x = origen_x
        self.__origen_y = origen_y
        self.__destino_x = destino_x
        self.__destino_y = destino_y
        self.__velocidad = velocidad
        self.__red = red
        self.__green = green
        self.__blue = blue
        """ Calculo de la distancia euclidiana """
        self.__distancia = distancia_euclidiana(
            origen_x, origen_y, destino_x, destino_y)
    """ Metodos geters """

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

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

    @property
    def origen_y(self):
        return self.__origen_y

    @property
    def destino_x(self):
        return self.__destino_x

    @property
    def destino_y(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):
        return self.__distancia

    def __str__(self):
        return (
            "#####################################\n"
            + "Id: " + str(self.__id) + ",\n"
            + "Origen X: " + str(self.__origen_x) + ",\n"
            + "Origen Y: " + str(self.__origen_y) + ",\n"
            + "Destino X: " + str(self.__destino_x) + ",\n"
            + "Destino Y: " + str(self.__destino_y) + ",\n"
            + "Velocidad: " + str(self.__velocidad) + ",\n"
            + "Rojo: " + str(self.__red) + ",\n"
            + "Verde: " + str(self.__green) + ",\n"
            + "Azul: " + str(self.__blue) + "\n")

    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
        }

    def __lt__(self, other):
        return self.id < other.id
```

Código archivo **listaParticula.py**

```
import json
from Particula import Particula
from algoritmos import distancia_euclidiana
```

```python
class listaParticula:
    def __init__(self):
        self.__particulas = []

    def agregar_inicio(self, particula: Particula):
        self.__particulas.insert(0, particula)

    def agregar_final(self, particula: Particula):
        self.__particulas.append(particula)

    def mostrar(self):
        for particula in self.__particulas:
            print(particula)

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

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

    def __len__(self):
        return len(self.__particulas)

    def __iter__(self):
        self.cont = 0
```

```
        return self

    def __next__(self):
        if self.cont < len(self.__particulas):
            """ Asignamos la particula a devolver """
            particula = self.__particulas[self.cont]
            """ Incremenamos el contador """
            self.cont += 1
            return particula
        """ detemos la iteración si se sobrepasa el tamaño de la lista """
        raise StopIteration

    def sortById(self):
        self.__particulas.sort()

    def sortByDistance(self):
        self.__particulas.sort(key=sort_distance, reverse=True)

    def sortBySpeed(self):
        self.__particulas.sort(key=sort_speed)

    """ Metodo de busqueda por fuerza bruta """

    def puntos_cercanos(self):
        """ Almacenará los resultados de la detección """
        resultado = []
        for punto_i in self.__particulas:
            x = punto_i.origen_x
            y = punto_i.origen_y
            """ utilizamos una distancia considerablemente grande para
forzar que el punto
             de comparación siempre tenga una distancia inferior al inicio,
despues se obtendrá
             el punto verdaderamente más cercano"""
            minimo = 1000
            cercano = (0, 0)
            color = (punto_i.red,
                     punto_i.green,
                     punto_i.blue)
            """ Segundo recorrido para hacer la comparación con el elemento
actual """
            for punto_j in self.__particulas:
                """ Evitamos comparar el punto inicial con el mismo """
                if (punto_i != punto_j):
                    """ Guardamos los nuevos puntos de comparación """
```

```
                    x_comparison = punto_j.origen_x
                    y_comparison = punto_j.origen_y
                    """ Obtenemos la distancia euclidiana entre los 2 puntos
para determinar si es
                        mayor o menor """
                    d = distancia_euclidiana(x, y, x_comparison,
y_comparison)
                    if d < minimo:
                        """ Corregimos el valor de la distancia minima para
proximas comparaciones """
                        minimo = d
                        """ Asignamos al punto más cercanos con los valores
de comparación actuales """
                        cercano = (x_comparison, y_comparison)
            """ Agregamos al arreglo el punto inicial de comparación y
depues el punto que se
            detectó como el más cercano """
            resultado.append(((x, y), cercano, color))
        """ Finalmente retornamos los resultados obtenidos """
        return resultado


""" Metodos fuera de "lista particula" """


def sort_distance(particula):
    return particula.distancia


def sort_speed(particula):
    return particula.velocidad
```

Código de archivo **algoritmos.py**

```
import math


def distancia_euclidiana(x_1, y_1, x_2, y_2):
    """ Devuelve el resultado de la distancia euclidiana """
    """
        x_1 -- origen x
        x_2 -- destino x
        y_1 -- origen y
        y-2 -- destino y
     """
```

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

Código de archivo **ui_mainwindow.py**

```
# -*- coding: utf-8 -*-

################################################################################
####
## Form generated from reading UI file 'mainwindow.ui'
##
## Created by: Qt User Interface Compiler version 5.15.2
##
## WARNING! All changes made in this file will be lost when recompiling UI
file!
################################################################################
####

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(779, 552)
        self.actionAbrir = QAction(MainWindow)
        self.actionAbrir.setObjectName(u"actionAbrir")
        self.actionGuardar = QAction(MainWindow)
        self.actionGuardar.setObjectName(u"actionGuardar")
        self.actionSortId = QAction(MainWindow)
        self.actionSortId.setObjectName(u"actionSortId")
        self.actionSortDistance = QAction(MainWindow)
        self.actionSortDistance.setObjectName(u"actionSortDistance")
        self.actionSortSpeed = QAction(MainWindow)
        self.actionSortSpeed.setObjectName(u"actionSortSpeed")
        self.actionSolo_puntos = QAction(MainWindow)
        self.actionSolo_puntos.setObjectName(u"actionSolo_puntos")
        self.actionOnlyOrigenXY = QAction(MainWindow)
        self.actionOnlyOrigenXY.setObjectName(u"actionOnlyOrigenXY")
        self.actionNearestPoints = QAction(MainWindow)
        self.actionNearestPoints.setObjectName(u"actionNearestPoints")
        self.centralwidget = QWidget(MainWindow)
```

```
        self.centralwidget.setObjectName(u"centralwidget")
        self.gridLayout_3 = QGridLayout(self.centralwidget)
        self.gridLayout_3.setObjectName(u"gridLayout_3")
        self.tabWidget = QTabWidget(self.centralwidget)
        self.tabWidget.setObjectName(u"tabWidget")
        self.tab = QWidget()
        self.tab.setObjectName(u"tab")
        self.gridLayout = QGridLayout(self.tab)
        self.gridLayout.setObjectName(u"gridLayout")
        self.groupBox = QGroupBox(self.tab)
        self.groupBox.setObjectName(u"groupBox")
        self.formLayout = QFormLayout(self.groupBox)
        self.formLayout.setObjectName(u"formLayout")
        self.label = QLabel(self.groupBox)
        self.label.setObjectName(u"label")

        self.formLayout.setWidget(4, QFormLayout.LabelRole, self.label)

        self.spnnDestinoX = QSpinBox(self.groupBox)
        self.spnnDestinoX.setObjectName(u"spnnDestinoX")
        self.spnnDestinoX.setMaximum(500)

        self.formLayout.setWidget(4, QFormLayout.FieldRole,
self.spnnDestinoX)

        self.label_2 = QLabel(self.groupBox)
        self.label_2.setObjectName(u"label_2")

        self.formLayout.setWidget(5, QFormLayout.LabelRole, self.label_2)

        self.spnnDestinoY = QSpinBox(self.groupBox)
        self.spnnDestinoY.setObjectName(u"spnnDestinoY")
        self.spnnDestinoY.setMaximum(500)

        self.formLayout.setWidget(5, QFormLayout.FieldRole,
self.spnnDestinoY)

        self.label_3 = QLabel(self.groupBox)
        self.label_3.setObjectName(u"label_3")

        self.formLayout.setWidget(6, QFormLayout.LabelRole, self.label_3)

        self.label_5 = QLabel(self.groupBox)
        self.label_5.setObjectName(u"label_5")
```

```
        self.formLayout.setWidget(7, QFormLayout.LabelRole, self.label_5)

        self.spnnRed = QSpinBox(self.groupBox)
        self.spnnRed.setObjectName(u"spnnRed")
        self.spnnRed.setMaximum(255)

        self.formLayout.setWidget(7, QFormLayout.FieldRole, self.spnnRed)

        self.label_6 = QLabel(self.groupBox)
        self.label_6.setObjectName(u"label_6")

        self.formLayout.setWidget(8, QFormLayout.LabelRole, self.label_6)

        self.spnnGreen = QSpinBox(self.groupBox)
        self.spnnGreen.setObjectName(u"spnnGreen")
        self.spnnGreen.setMaximum(255)

        self.formLayout.setWidget(8, QFormLayout.FieldRole, self.spnnGreen)

        self.label_7 = QLabel(self.groupBox)
        self.label_7.setObjectName(u"label_7")

        self.formLayout.setWidget(9, QFormLayout.LabelRole, self.label_7)

        self.spnnBlue = QSpinBox(self.groupBox)
        self.spnnBlue.setObjectName(u"spnnBlue")
        self.spnnBlue.setMaximum(255)

        self.formLayout.setWidget(9, QFormLayout.FieldRole, self.spnnBlue)

        self.btnAgregarFinal = QPushButton(self.groupBox)
        self.btnAgregarFinal.setObjectName(u"btnAgregarFinal")
        self.btnAgregarFinal.setCursor(QCursor(Qt.PointingHandCursor))

        self.formLayout.setWidget(13, QFormLayout.SpanningRole,
self.btnAgregarFinal)

        self.label_4 = QLabel(self.groupBox)
        self.label_4.setObjectName(u"label_4")

        self.formLayout.setWidget(0, QFormLayout.LabelRole, self.label_4)

        self.spnnOrigenX = QSpinBox(self.groupBox)
        self.spnnOrigenX.setObjectName(u"spnnOrigenX")
        self.spnnOrigenX.setMaximum(500)
```

```
        self.formLayout.setWidget(0, QFormLayout.FieldRole,
self.spnnOrigenX)

        self.spnnOrigenY = QSpinBox(self.groupBox)
        self.spnnOrigenY.setObjectName(u"spnnOrigenY")
        self.spnnOrigenY.setMaximum(500)

        self.formLayout.setWidget(2, QFormLayout.FieldRole,
self.spnnOrigenY)

        self.label_8 = QLabel(self.groupBox)
        self.label_8.setObjectName(u"label_8")

        self.formLayout.setWidget(2, QFormLayout.LabelRole, self.label_8)

        self.spnnVelocidad = QSpinBox(self.groupBox)
        self.spnnVelocidad.setObjectName(u"spnnVelocidad")
        self.spnnVelocidad.setMaximum(500)

        self.formLayout.setWidget(6, QFormLayout.FieldRole,
self.spnnVelocidad)

        self.btnAgregarInicio = QPushButton(self.groupBox)
        self.btnAgregarInicio.setObjectName(u"btnAgregarInicio")
        self.btnAgregarInicio.setCursor(QCursor(Qt.PointingHandCursor))

        self.formLayout.setWidget(14, QFormLayout.SpanningRole,
self.btnAgregarInicio)

        self.btnMostrar = QPushButton(self.groupBox)
        self.btnMostrar.setObjectName(u"btnMostrar")
        self.btnMostrar.setCursor(QCursor(Qt.PointingHandCursor))

        self.formLayout.setWidget(15, QFormLayout.SpanningRole,
self.btnMostrar)

        self.label_9 = QLabel(self.groupBox)
        self.label_9.setObjectName(u"label_9")

        self.formLayout.setWidget(10, QFormLayout.LabelRole, self.label_9)

        self.spnnDistancia = QSpinBox(self.groupBox)
        self.spnnDistancia.setObjectName(u"spnnDistancia")
        self.spnnDistancia.setMaximum(255)
```

```
        self.formLayout.setWidget(10, QFormLayout.FieldRole,
self.spnnDistancia)


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

        self.plainTextEdit = QPlainTextEdit(self.tab)
        self.plainTextEdit.setObjectName(u"plainTextEdit")
        self.plainTextEdit.setMaximumSize(QSize(500, 16777215))
        self.plainTextEdit.setFrameShadow(QFrame.Raised)

        self.gridLayout.addWidget(self.plainTextEdit, 0, 1, 1, 1)

        self.tabWidget.addTab(self.tab, "")
        self.tab_2 = QWidget()
        self.tab_2.setObjectName(u"tab_2")
        self.gridLayout_2 = QGridLayout(self.tab_2)
        self.gridLayout_2.setObjectName(u"gridLayout_2")
        self.tableParticulas = QTableWidget(self.tab_2)
        self.tableParticulas.setObjectName(u"tableParticulas")

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

        self.searchEdit = QLineEdit(self.tab_2)
        self.searchEdit.setObjectName(u"searchEdit")

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

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

        self.gridLayout_2.addWidget(self.buscar_pushButton, 1, 1, 1, 1)

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

        self.gridLayout_2.addWidget(self.mostrar_pushButton, 1, 2, 1, 1)

        self.tabWidget.addTab(self.tab_2, "")
        self.tab_3 = QWidget()
        self.tab_3.setObjectName(u"tab_3")
        self.gridLayout_4 = QGridLayout(self.tab_3)
        self.gridLayout_4.setObjectName(u"gridLayout_4")
        self.dibujarPushButton = QPushButton(self.tab_3)
```

```
        self.dibujarPushButton.setObjectName(u"dibujarPushButton")

        self.gridLayout_4.addWidget(self.dibujarPushButton, 2, 0, 1, 1)

        self.limpiarPushBtn = QPushButton(self.tab_3)
        self.limpiarPushBtn.setObjectName(u"limpiarPushBtn")

        self.gridLayout_4.addWidget(self.limpiarPushBtn, 2, 1, 1, 1)

        self.particulasView = QGraphicsView(self.tab_3)
        self.particulasView.setObjectName(u"particulasView")

        self.gridLayout_4.addWidget(self.particulasView, 0, 0, 1, 2)

        self.tabWidget.addTab(self.tab_3, "")

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

        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QMenuBar(MainWindow)
        self.menubar.setObjectName(u"menubar")
        self.menubar.setGeometry(QRect(0, 0, 779, 26))
        self.menuArchivo = QMenu(self.menubar)
        self.menuArchivo.setObjectName(u"menuArchivo")
        self.menuOrdenar = QMenu(self.menubar)
        self.menuOrdenar.setObjectName(u"menuOrdenar")
        self.menuCercan_a = QMenu(self.menubar)
        self.menuCercan_a.setObjectName(u"menuCercan_a")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QStatusBar(MainWindow)
        self.statusbar.setObjectName(u"statusbar")
        MainWindow.setStatusBar(self.statusbar)

        self.menubar.addAction(self.menuArchivo.menuAction())
        self.menubar.addAction(self.menuOrdenar.menuAction())
        self.menubar.addAction(self.menuCercan_a.menuAction())
        self.menuArchivo.addAction(self.actionAbrir)
        self.menuArchivo.addAction(self.actionGuardar)
        self.menuOrdenar.addAction(self.actionSortId)
        self.menuOrdenar.addAction(self.actionSortDistance)
        self.menuOrdenar.addAction(self.actionSortSpeed)
        self.menuCercan_a.addAction(self.actionOnlyOrigenXY)
        self.menuCercan_a.addAction(self.actionNearestPoints)

        self.retranslateUi(MainWindow)
```

```
        self.tabWidget.setCurrentIndex(1)


        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+G", None))
#endif // QT_CONFIG(shortcut)
        self.actionSortId.setText(QCoreApplication.translate("MainWindow",
u"Por id (ascendente)...", None))
#if QT_CONFIG(shortcut)
        self.actionSortId.setShortcut(QCoreApplication.translate("MainWindow
", u"Shift+I", None))
#endif // QT_CONFIG(shortcut)
        self.actionSortDistance.setText(QCoreApplication.translate("MainWind
ow", u"Por distancia (descendente)...", None))
#if QT_CONFIG(shortcut)
        self.actionSortDistance.setShortcut(QCoreApplication.translate("Main
Window", u"Shift+D", None))
#endif // QT_CONFIG(shortcut)
        self.actionSortSpeed.setText(QCoreApplication.translate("MainWindow"
, u"Velocidad (ascendente)...", None))
#if QT_CONFIG(shortcut)
        self.actionSortSpeed.setShortcut(QCoreApplication.translate("MainWin
dow", u"Shift+V", None))
#endif // QT_CONFIG(shortcut)
        self.actionSolo_puntos.setText(QCoreApplication.translate("MainWindo
w", u"Origen", None))
        self.actionOnlyOrigenXY.setText(QCoreApplication.translate("MainWind
ow", u"Solo origen x-y", None))
#if QT_CONFIG(shortcut)
```

```
        self.actionOnlyOrigenXY.setShortcut(QCoreApplication.translate("Main
Window", u"Ctrl+Shift+I", None))
#endif // QT_CONFIG(shortcut)
        self.actionNearestPoints.setText(QCoreApplication.translate("MainWin
dow", u"Puntos m\u00e1s cercanos", None))
#if QT_CONFIG(shortcut)
        self.actionNearestPoints.setShortcut(QCoreApplication.translate("Mai
nWindow", u"Ctrl+Shift+N", None))
#endif // QT_CONFIG(shortcut)
        self.groupBox.setTitle(QCoreApplication.translate("MainWindow",
u"GroupBox", None))
        self.label.setText(QCoreApplication.translate("MainWindow",
u"Destino X", None))
        self.label_2.setText(QCoreApplication.translate("MainWindow",
u"Destino Y", None))
        self.label_3.setText(QCoreApplication.translate("MainWindow",
u"Velocidad", None))
        self.label_5.setText(QCoreApplication.translate("MainWindow",
u"Red", None))
        self.label_6.setText(QCoreApplication.translate("MainWindow",
u"Green", None))
        self.label_7.setText(QCoreApplication.translate("MainWindow",
u"Blue", None))
        self.btnAgregarFinal.setText(QCoreApplication.translate("MainWindow"
, u"Agregar final", None))
        self.label_4.setText(QCoreApplication.translate("MainWindow",
u"Origen X:", None))
        self.label_8.setText(QCoreApplication.translate("MainWindow",
u"Origen Y: ", None))
        self.btnAgregarInicio.setText(QCoreApplication.translate("MainWindow
", u"Agregar inicio", None))
        self.btnMostrar.setText(QCoreApplication.translate("MainWindow",
u"Mostrar", None))
        self.label_9.setText(QCoreApplication.translate("MainWindow",
u"Distancia:", None))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab),
QCoreApplication.translate("MainWindow", u"Agregar", None))
        self.searchEdit.setPlaceholderText(QCoreApplication.translate("MainW
indow", u"Id de particula", None))
        self.buscar_pushButton.setText(QCoreApplication.translate("MainWindo
w", u"Buscar", None))
        self.mostrar_pushButton.setText(QCoreApplication.translate("MainWind
ow", u"Mostrar", None))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab_2),
QCoreApplication.translate("MainWindow", u"Tabla", None))
```

```
        self.dibujarPushButton.setText(QCoreApplication.translate("MainWindo
w", u"Dibujar", None))
        self.limpiarPushBtn.setText(QCoreApplication.translate("MainWindow",
u"Limpiar", None))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab_3),
QCoreApplication.translate("MainWindow", u"Page", None))
        self.menuArchivo.setTitle(QCoreApplication.translate("MainWindow",
u"Archivo", None))
        self.menuOrdenar.setTitle(QCoreApplication.translate("MainWindow",
u"Ordenar", None))
        self.menuCercan_a.setTitle(QCoreApplication.translate("MainWindow",
u"Cercan\u00eda", None))
    # retranslateUi
```

Código de subida al git.

```
Armando@Armando04 MINGW64 ~/Documents/GitHub/Actividad 11 (main)
$ git init
Initialized empty Git repository in C:/Users/Armando/Documents/GitHub/Actividad
11/.git/

Armando@Armando04 MINGW64 ~/Documents/GitHub/Actividad 11 (master)
$ git add .

Armando@Armando04 MINGW64 ~/Documents/GitHub/Actividad 11 (master)
$ git commit -m "Subida de la actividad 11"
[master (root-commit) ce2c451] Subida de la actividad 11
 13 files changed, 1725 insertions(+)
 create mode 100644 particulas 2.json
 create mode 100644 src/Main.py
 create mode 100644 src/Particula.py
 create mode 100644 src/__pycache__/Particula.cpython-310.pyc
 create mode 100644 src/__pycache__/algoritmos.cpython-310.pyc
 create mode 100644 src/__pycache__/listaParticulas.cpython-310.pyc
 create mode 100644 src/__pycache__/mainwindow.cpython-310.pyc
 create mode 100644 src/__pycache__/ui_mainwindow.cpython-310.pyc
 create mode 100644 src/algoritmos.py
 create mode 100644 src/listaParticulas.py
 create mode 100644 src/mainwindow.py
 create mode 100644 src/mainwindow.ui
 create mode 100644 src/ui_mainwindow.py

Armando@Armando04 MINGW64 ~/Documents/GitHub/Actividad 11 (master)
$ git branch -M main

Armando@Armando04 MINGW64 ~/Documents/GitHub/Actividad 11 (main)
$ git remote add origin https://github.com/Diego-Armando-H/Actividad-11.git

Armando@Armando04 MINGW64 ~/Documents/GitHub/Actividad 11 (main)
$ git push -u origin main
Enumerating objects: 17, done.
Counting objects: 100% (17/17), done.
Delta compression using up to 4 threads
Compressing objects: 100% (17/17), done.
Writing objects: 100% (17/17), 19.32 KiB | 2.42 MiB/s, done.
Total 17 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Diego-Armando-H/Actividad-11.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```