

# **Actividad 11:Fuerza bruta**

**Ramirez Orduna Geovanni**

## **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.

Tenemos las partículas agregadas

MainWindow

Archivo

Agregar

Tabla

Particulas

ID

5

Origen X

54

Origen Y

6

Destino x:

4

Destino y:

5

Velocidad:

45

Rojo:

1

Verde:

3

Azul:

3

Agregar Final

Agregar inicio

Mostrar

Ordenar

id: 5

origen x: 54

origen y: 6

destino x: 4

destino y: 5

velocidad: 45

red: 1

green: 3

blue: 3

distancia: 48.010415536631214

id: 3

origen x: 3

origen y: 5

destino x: 4

destino y: 5

velocidad: 45

red: 1

green: 3

blue: 3

distancia: 2.23606797749979

MainWindow

Archivo

Agregar Tabla

	id	origen x	origen y	destono x	destino y	velocidad	red	green	blue	distancia
1	5	54	6	4	5	45	1	3	3	48.010415...
2	3	3	5	4	5	45	1	3	3	2....

3

Ordenar Buscar Mostrar

Tenemos los datos mostrados en la tabla

Conclusión: En esta práctica pude conocer cómo se realiza el método sort para así poder ordenar de forma ascendente como descendente.

Referencias :

<https://www.youtube.com/watch?v=0NZajLly5qQ&t=1037s>

codigo

```
from PySide2.QtWidgets import QMainWindow, QGraphicsScene
from ui_mainwindow import Ui_MainWindow
from PySide2.QtGui import QPen, QColor, QTransform
from PySide2.QtCore import Slot
from random import randint

class MainWindow(QMainWindow):
    def __init__(self):
        super(MainWindow, self).__init__()
        self.ui=Ui_MainWindow()
        self.ui.setupUi(self)

        self.ui.Draw_Button.clicked.connect(self.draw)
        self.ui.Clean_Buttom.clicked.connect(self.clean)

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

    def wheelEvent(self, event):
        if event.delta() >0:
            self.ui.graphicsView.scale(1.2,1.2)
        else:
            self.ui.graphicsView.scale(0.8,0.8)

    @Slot()
    def draw(self):
        pen=QPen()
        pen.setWidth(2)
        for i in range(3):
            r=randint(0,255)
            g=randint(0,255)
            b=randint(0,255)
            color=QColor(r,g,b)

            origen_x=randint(0,500)
            origen_y=randint(0,500)
            destino_x=randint(0,500)
```

```
destino_y=randint(0,500)

pen.setColor(color)
self.scene.addEllipse(origen_x,origen_y,3,3,pen)
self.scene.addEllipse(destino_x,destino_y,3,3,pen)
self.scene.addLine(origen_x+3,
origen_y+3,destino_x,destino_y,pen)

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

mainWindow

```
import PySide2
from PySide2.QtWidgets import QMainWindow, QFileDialog, QMessageBox
from PySide2.QtCore import Slot
from ui_mainwindow import Ui_MainWindow
from admin import Admin
from particula import Particula
from algoritmos import distancia_euclidiana

class MainWIndow(QMainWindow):
    def __init__(self):
        super(MainWIndow, self).__init__()

        self.admin=Admin()

        self.ui=Ui_MainWindow()
        self.ui.setupUi(self) ##mete la intrerfaz
        self.ui.Agre_Final_button.clicked.connect(self.click_agregar)
##conectar clase a boton

self.ui.Agre_Inicio_Button.clicked.connect(self.click_agregar_inicio)
        self.ui.Mostrar_button.clicked.connect(self.click_mostrar)


        self.ui.actionAbrir.triggered.connect(self.action_abrir_archivo)
        self.ui.actionGuardar.triggered.connect(self.action_guardar)


    @Slot()
    def action_abrir_archivo(self):
        #print("abrir archivo")
        ubicacion=QFileDialog.getOpenFileName(
            self,
            "abrir archivo",
            ".",
            "JSON (*.json)"
        )[0]
        if self.admin.open(ubicacion):
            QMessageBox.information(
                self,
                "exito",
                "Se abrio el archivo"+ubicacion
            )
```

```

        else:
            QMessageBox.critical(
                self,
                "Error",
                "Error abrir archivo "
            )

@Slot()
def action_guardar(self):
    #print("guardar")
    ubicacion= QFileDialog.getSaveFileName(
        self,
        'Guardar Archivo',
        '.',
        'JSON (*.json)'
    )[0]
    print(ubicacion)
    self.admin.save(ubicacion)
    if self.admin.save(ubicacion):
        QMessageBox.information(
            self,
            "exito",
            "se pudo crear archivo"+ubicacion
        )
    else:
        QMessageBox.critical(
            self,
            "Error",
            "no se pudo crear archivo"
        )

@Slot()
def click_agregar(self):
    id=self.ui.Id_spinBox.value()
    OrigenX=self.ui.OrigenX_spinBox.value()
    OrigenY=self.ui.OrigenY_spinBox.value()
    DestinoX=self.ui.DestinoX_spinBox.value()
    DestinoY=self.ui.DestinoY_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()
        distancia=distacia_euclidiana

particula=Particula(id,OrigenX,OrigenY,DestinoX,DestinoY,Velocidad,
Red,Green,Blue,distancia)
        self.admin.agregar_final(particula)

        #print(OrigenX,OrigenY,
DestinoX,DestinoY,Velocidad,Red,Green,Blue)
        #self.ui.salida.insertPlainText(str(OrigenX) + str( OrigenY)
+ str(DestinoX) + str (DestinoY) + str (Velocidad) +str( Red)+
str(Green) + str(Blue))
    @Slot()
    def click_agregar_inicio(self):
        id=self.ui.Id_spinBox.value()
        OrigenX=self.ui.OrigenX_spinBox.value()
        OrigenY=self.ui.OrigenY_spinBox.value()
        DestinoX=self.ui.DestinoX_spinBox.value()
        DestinoY=self.ui.DestinoY_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()
        distancia=distacia_euclidiana

particula=Particula(id,OrigenX,OrigenY,DestinoX,DestinoY,Velocidad,
Red,Green,Blue,distancia)
        self.admin.agregar_inicio(particula)

    @Slot()
    def click_mostrar(self):
        # self.admin.mostrar()
        self.ui.salida.clear()
        self.ui.salida.insertPlainText(str(self.admin))

```