Actividad 11 – Fuerza Bruta

Cervantes Torres Carlos Alberto

Seminario de solución de problemas de algoritmia

Lineamientos de evaluación

Aquí deberás escribir los lineamientos de evaluación descritos en la actividad, señalando o marcando aquellos lineamientos que estás cumpliendo

[x] El reporte está en formato Google Docs o PDF.

[x] El reporte sigue las pautas del Formato de Actividades .

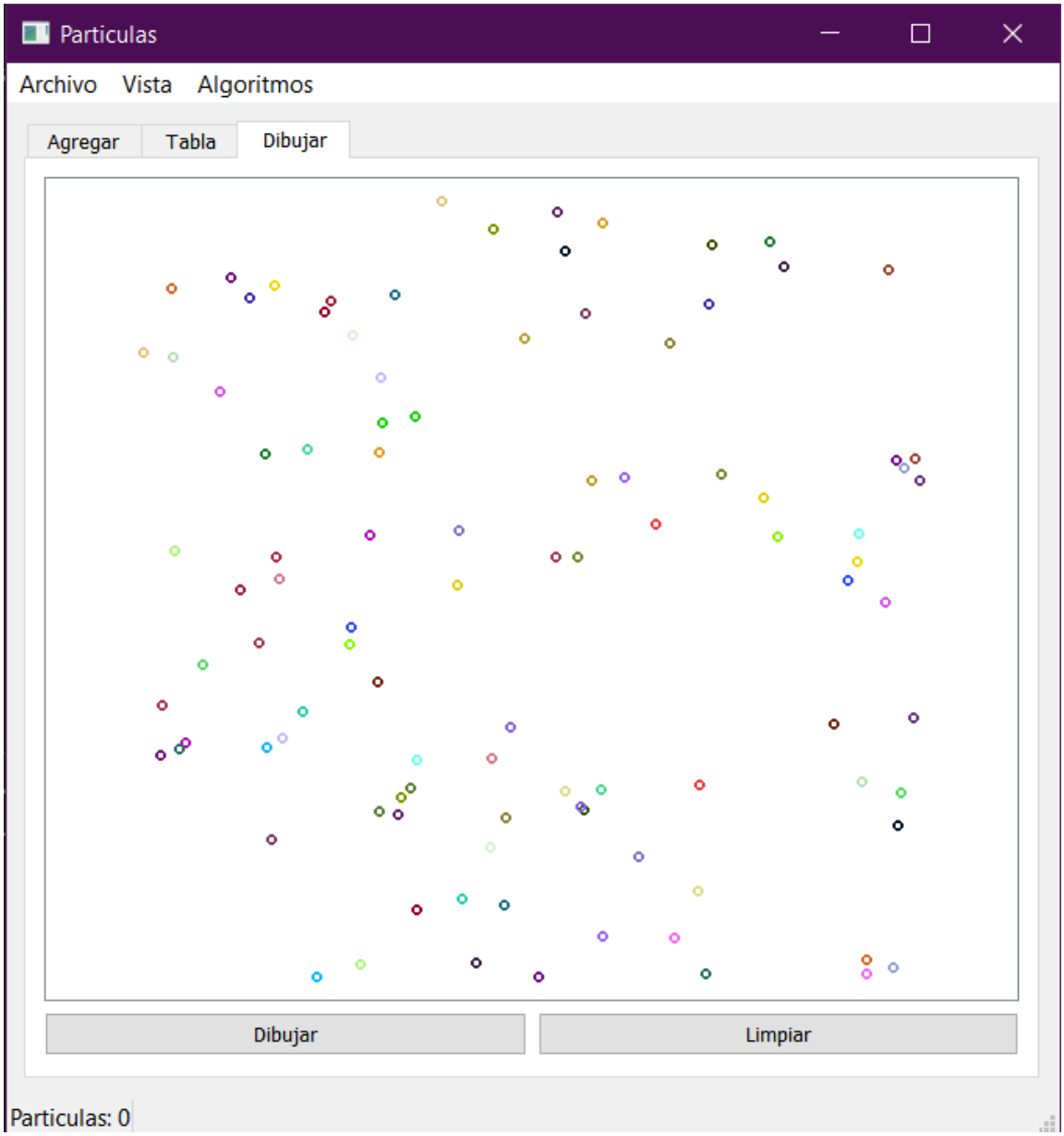
[x] El reporte tiene desarrollada todas las pautas del Formato de Actividades.

[ ] Se muestra captura de pantalla de los puntos de las partículas en el QScene.

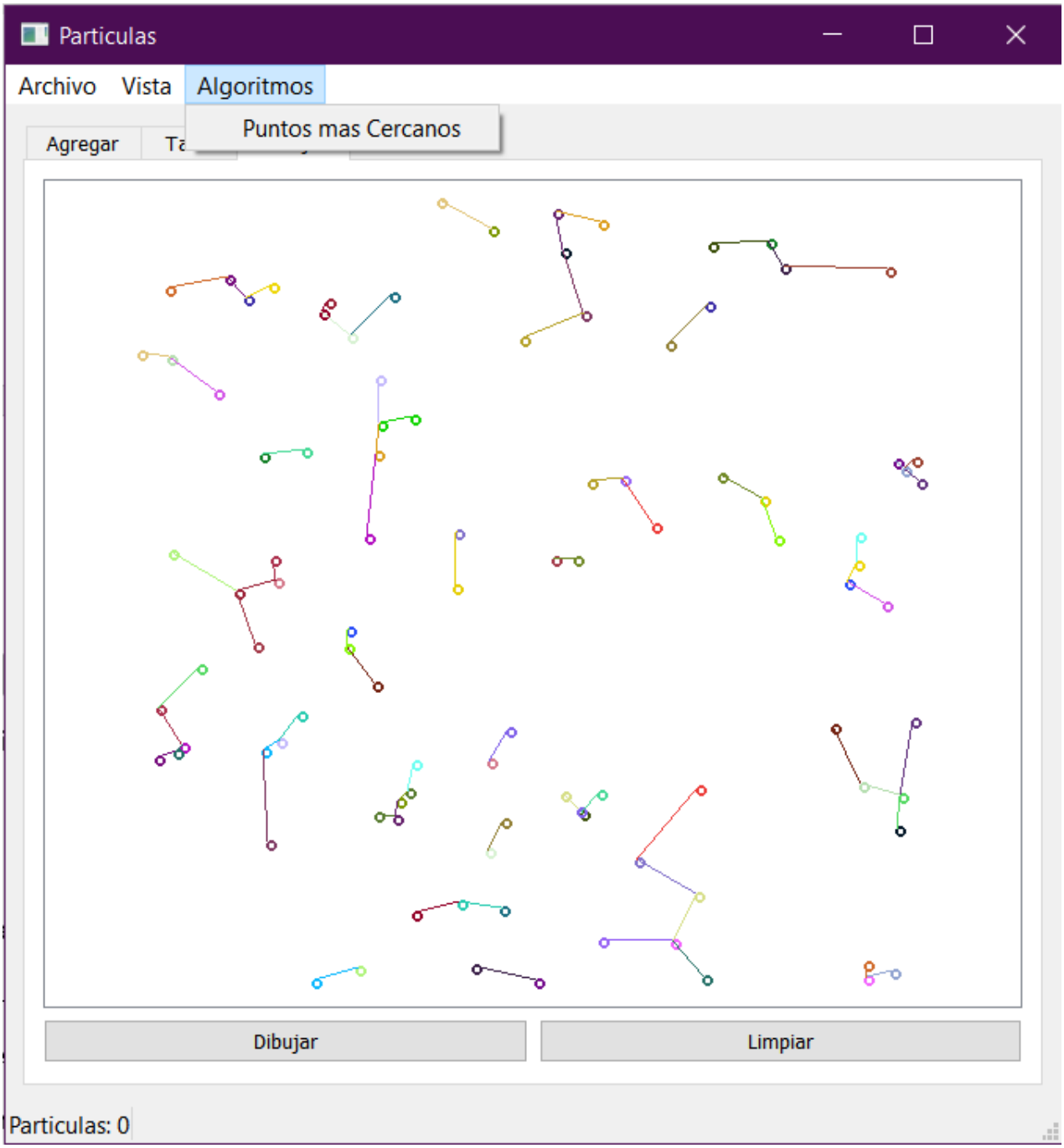
[ ] Se muestra captura de pantalla del resultado del algoritmo de fuerza bruta en el QScene.

Desarrollo

Puntos de las particulas en el QScene



Algoritmo de Fuerza Bruta en el QScene



Conclusiones

En esta actividad me complique bastante pero al final lo logre al principio hice el algoritmo en algoritmos pero como me tarde un poco lo cambie todo agregue la función get_puntos en la clase lista o clase administradora de particulas y en ese archivo cree una clase Punto para que me diera los puntos y por ultimo cree las funciones de dibujar_puntos, calcular_puntos_mas_cercanos y dibujar_puntos_cercanos en el mainwindow conectados con el trigger de dibujar los puntos y para el algoritmo de fuerza bruta de los puntos mas cercanos. Y fue mas complicado el saber en donde crear las funciones pero ya por fin quedo el algoritmo.

Referencias

https://youtu.be/hPARP71VEH0 , Actividad 11 (Fuerza Bruta), Michel Davalos Boites.

Código

mainwindow.py

```
from PySide2.QtWidgets import QMainWindow, QFileDialog, QMessageBox,
QTableWidgetItem, QGraphicsScene, QLabel
from PySide2.QtCore import Slot
from PySide2.QtGui import QPen, QColor, QTransform
from ui_mainwindow import Ui_MainWindow
from lista import Lista
from particula import Particula
from algoritmos import distancia_euclidiana
from lista import Punto

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

        self.lista = Lista()
```

```
        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)
        self.label = QLabel()
        self.ui.statusbar.addWidget(self.label)
        #conexion con los botones de la interfaz
        self.ui.agregar_final_pushButton.clicked.connect(self.click_agregar)
        self.ui.agregar_inicio_pushButton.clicked.connect(self.click_agregar_inici
o)
        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.mostar_tabla)
        self.ui.buscar_pushButton.clicked.connect(self.buscar_id)
        #Dibujar y limpiar
        self.ui.dibujar.clicked.connect(self.dibujar)
        self.ui.limpiar.clicked.connect(self.limpiar)

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

        self.ui.ordenar_id_pushButton.clicked.connect(self.ordenar_id)
        self.ui.ordenar_distancia_pushButton.clicked.connect(self.ordenar_distanci
a)
        self.ui.ordenar_velocidad_pushButton.clicked.connect(self.ordenar_velocida
d)


        self.puntos = []
        self.puntos_cercanos = []
        self.ui.actionPuntos.triggered.connect(self.dibujar_puntos)
        self.ui.actionPuntos_mas_Cercanos.triggered.connect(self.calcular_puntos_m
as_cercanos)
        self.actualizar_particulas()

    #Funcion puntos cercanos
    @Slot()
    def calcular_puntos_mas_cercanos(self):
```

```
        for punto01 in self.puntos:
            distMin = 1000
            punto = Punto()
            for todos in self.puntos:
                if punto01 == todos:
                    continue
                dist = distancia_euclidiana(punto01.x, punto01.y, todos.x,
todos.y)
                if dist < distMin:
                    distMin = dist
                    punto = todos
            self.puntos_cercanos.append([punto01, punto])
        self.dibujar_puntos_mas_cercanos()


    def dibujar_puntos_mas_cercanos(self):
        for punto01, punto02 in self.puntos_cercanos:
            pen = QPen()
            color = QColor(punto01.red, punto01.green, punto01.blue)
            pen.setColor(color)
            self.scene.addLine(punto01.x, punto01.y, punto02.x, punto02.y, pen)

    #Funcion actualizar
    @Slot()
    def actualizar_particulas(self):
        self.label.setText(f"Particulas: {self.lista.cantidad()}")
    #Funcion para dibujar los puntos conectada con el trigger
    @Slot()
    def dibujar_puntos(self):
        self.puntos = self.lista.get_puntos()
        for punto in self.puntos:
            x = punto.x
            y = punto.y
            red = punto.red
            green = punto.green
            blue = punto.blue
            color = QColor(red, green, blue)
            pen = QPen()
```

```
            pen.setWidth(2)
            pen.setColor(color)

            self.scene.addEllipse(x, y, 5, 5, pen)
```

lista.py

```
def get_puntos(self):
        puntos = []

        for particula in self.__particulas:
            punto01 = Punto(particula.origen_x, particula.origen_y, particula.red,
particula.green, particula.blue)
            punto02 = Punto(particula.destino_x, particula.destino_y,
particula.red, particula.green, particula.blue)

            puntos.append(punto01)
            puntos.append(punto02)
        return puntos

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

class Punto:
    def __init__(self, x:int=0, y:int=0, red:int=0, green:int=0, blue:int=0):
        self.x = x
        self.y = y
        self.red = red
        self.green = green
        self.blue = blue
```