Seminario de Algoritmia

CLAVE: I59556

NRC: 59556

2022B

D14

UI User Interface

Arellano Granados Angel Mariano

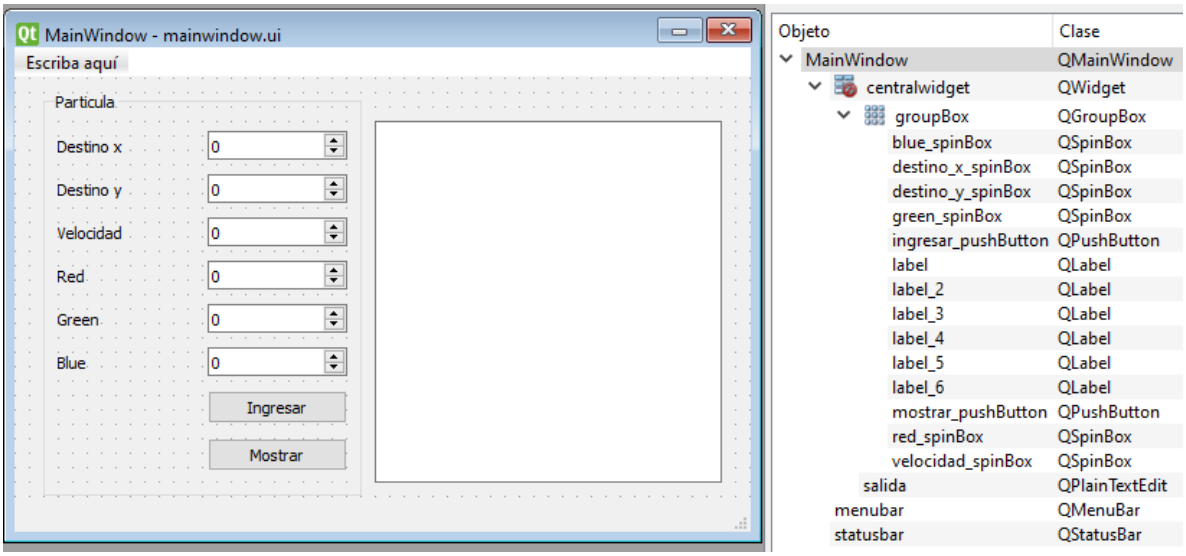
218123444

Descripción de la Actividad:

Usando la herramienta de creación interfaces de usuario QT designer crearemos una UI que enlazaremos a un programa de Python que permitirá capturar y mostrar registros de partículas.

Contenido de la Actividad:

Creación del UI en QT designer:



Clase partícula:

particula.py

```python
class Particula:
    def __init__(self, destino_x="", destino_y="", velocidad="",red=0,green=0,blue=0):
        self.__destino_x = destino_x
        self.__destino_y = destino_y
        self.__velocidad = velocidad
        self.__red = red
        self.__green = green
        self.__blue = blue

    def __str__(self):
        return(
            "Destino x: "+ str(self.__destino_x) + "\n"+
            "Destino y: " + str(self.__destino_y) + "\n"+
            "Velocidad: " + str(self.__velocidad) + "\n"+
            "Red: " + str(self.__red) + "\n"+
            "Green: " + str(self.__green) + "\n"+
            "Blue: " + str(self.__blue) + "\n"
        )
```

Clase administradora:

admin.py

```python
from particula import Particula

class Admin:
    def __init__(self):
        self.__particulas = []

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

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

    def __str__(self) -> str:
        return"".join(
            str(v) + "\n" for v in self.__particulas
        )
```

Mianwindow.py:

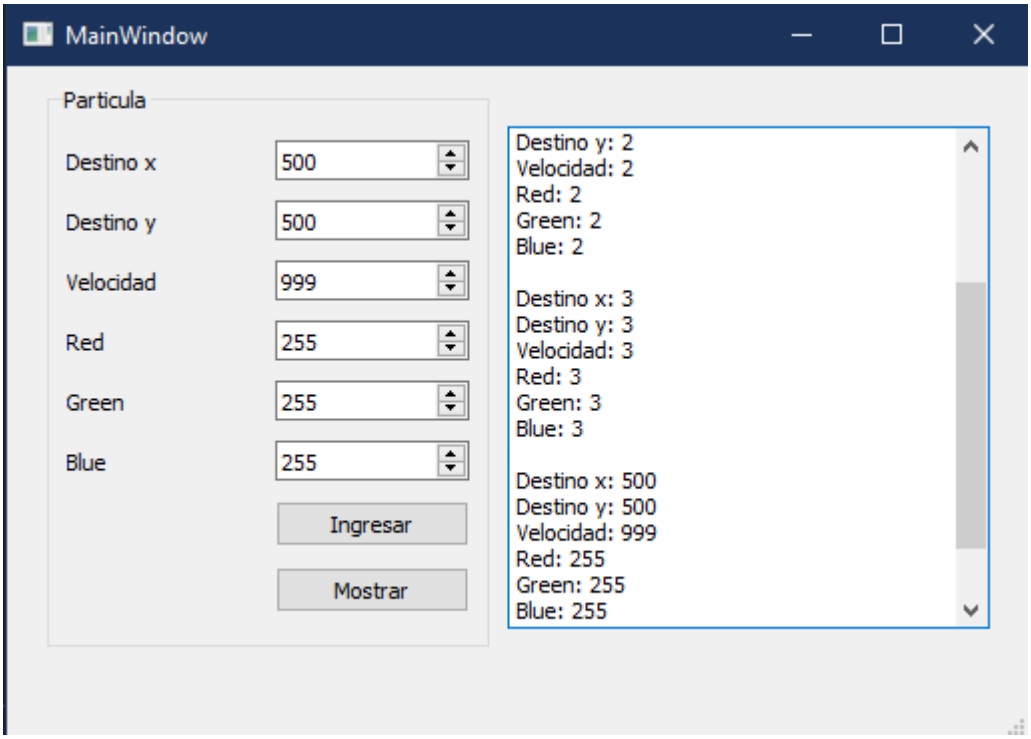
mainwindow.py 2

```python
from PySide2.QtWidgets import QMainWindow
from PySide2.QtCore import Slot
from ui_mainwindow import Ui_MainWindow
from admin import Admin
from particula import Particula

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

        self.admin = Admin()

        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)

        self.ui.ingresar_pushButton.clicked.connect(self.click_agregar)
        self.ui.mostrar_pushButton.clicked.connect(self.click_mostrar)

    @Slot( )
    def click_agregar(self):
        destino_x = self.ui.destino_x_spinBox.value()
        destino_y = self.ui.destino_y_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()

        particula = Particula(destino_x,destino_y,velocidad,red,green,blue)
        self.admin.agrega_final(particula)

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

Main.py:

```python
from PySide2.QtWidgets import QMainWindow, QApplication
from mainwindow import MainWindow
import sys

#APlicacion de Qt
app = QApplication()

#Se crea una ventana vacia
window = MainWindow()

#Se hace visible la ventana
window.show()

#Qt loop
sys.exit(app.exec_())
```

Prueba de ejecución:



Conclusión:

Esta actividad resulto fácil ya que solo era iterar con los códigos y recursos que habíamos ido juntando a lo largo de las clases, aun asi hubo cosas nuevas como imitar el alcance de los spinBox para que alcanzaran los valores requeridos.