UNIVERSIDAD AUTÓNOMA DE CHIAPAS.

FACULTAD DE CONTADURÍA Y ADMINISTRACIÓN, CAMPUS I.

LICENCIATURA EN INGENIERÍA EN DESARROLLO Y TECNOLOGÍAS DE SOFTWARE.

OCTAVO SEMESTRE, GRUPO: “M”

MATERIA: GRAFICACION.

DOCENTE: MTRO. SANDOVAL ZUÑIGA LUIS MANUEL.

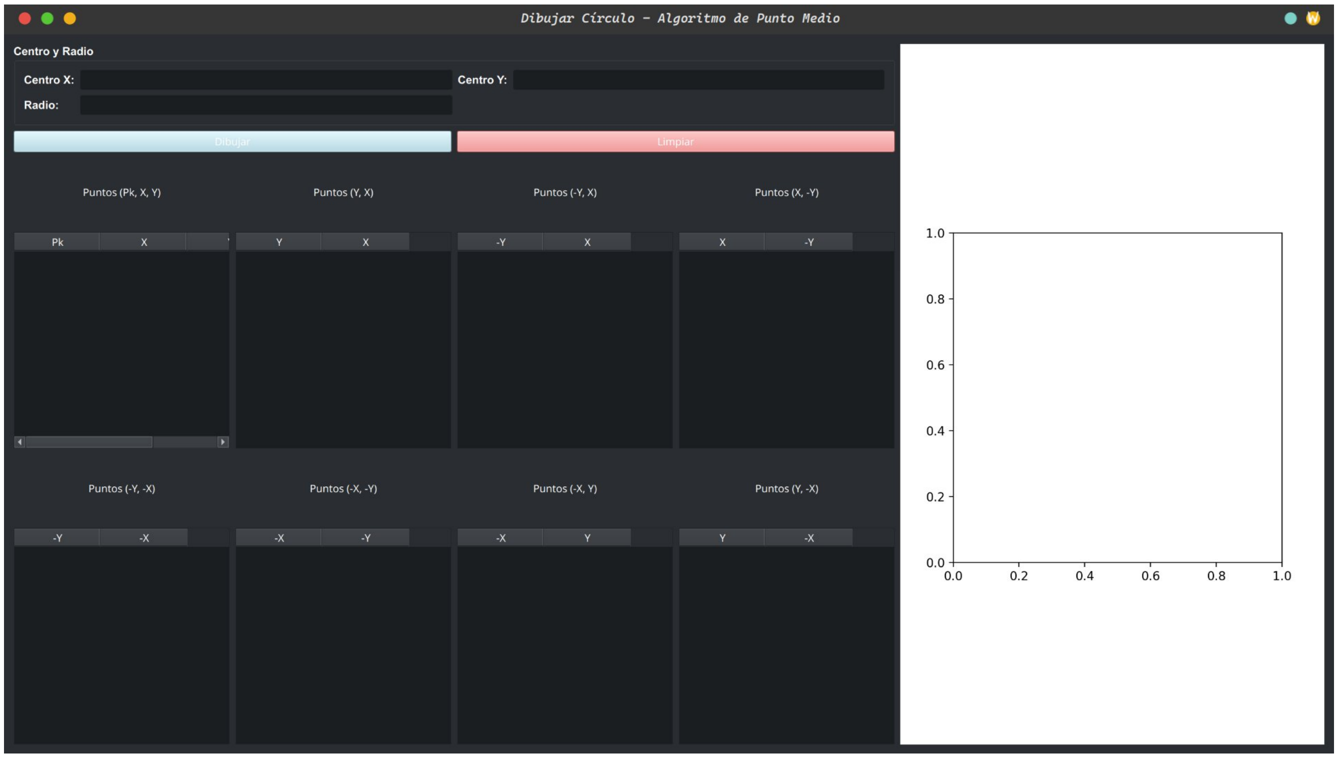
ALUMNOS:

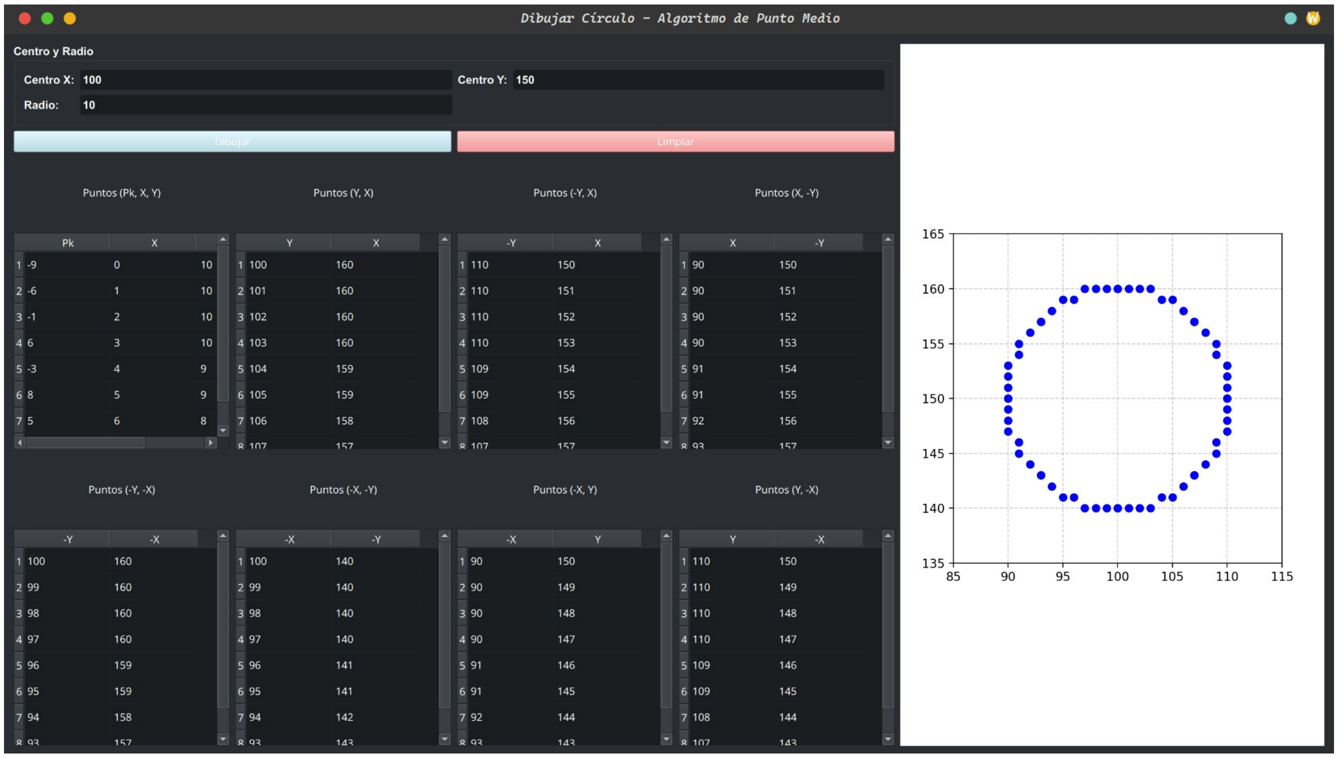
- CARLOS DANIEL AMORES HERNANDEZ – A210367
- CRISTOBAL DE JESUS CORONEL CHAMBE – A210016
- JESUS ADRIAN CRUZ LEON – A210395

“3er. DOCUMENTO PROGRAMA CIRCULO RELLENO”

FECHA DE ENTREGA: 22 DE MARZO DEL 2025.

CAPTURA DE PANTALLA DEL PROGRAMA





Centro X: 100
Centro Y: 150
Radio: 10

Puntos (Pk, X, Y)

	Pk	X	
1	-9	0	10
2	-6	1	10
3	-1	2	10
4	6	3	10
5	-3	4	9
6	8	5	9
7	5	6	8

Puntos (Y, X)

	Y	X
1	100	160
2	101	160
3	102	160
4	103	160
5	104	159
6	105	159
7	106	158
8	107	157

Puntos (-Y, X)

	-Y	X
1	110	150
2	110	151
3	110	152
4	110	153
5	109	154
6	109	155
7	108	156
8	107	157

Puntos (X, -Y)

	X	-Y
1	90	150
2	90	151
3	90	152
4	90	153
5	91	154
6	91	155
7	92	156
8	93	157

Puntos (-Y, -X)

	-Y	-X
1	100	160
2	99	160
3	98	160
4	97	160
5	96	159
6	95	159
7	94	158
8	93	157

Puntos (-X, -Y)

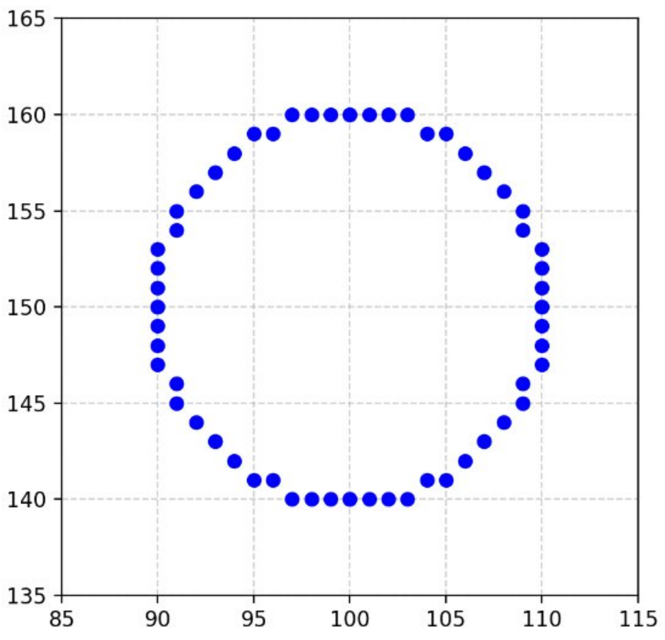
	-X	-Y
1	100	140
2	99	140
3	98	140
4	97	140
5	96	141
6	95	141
7	94	142
8	93	143

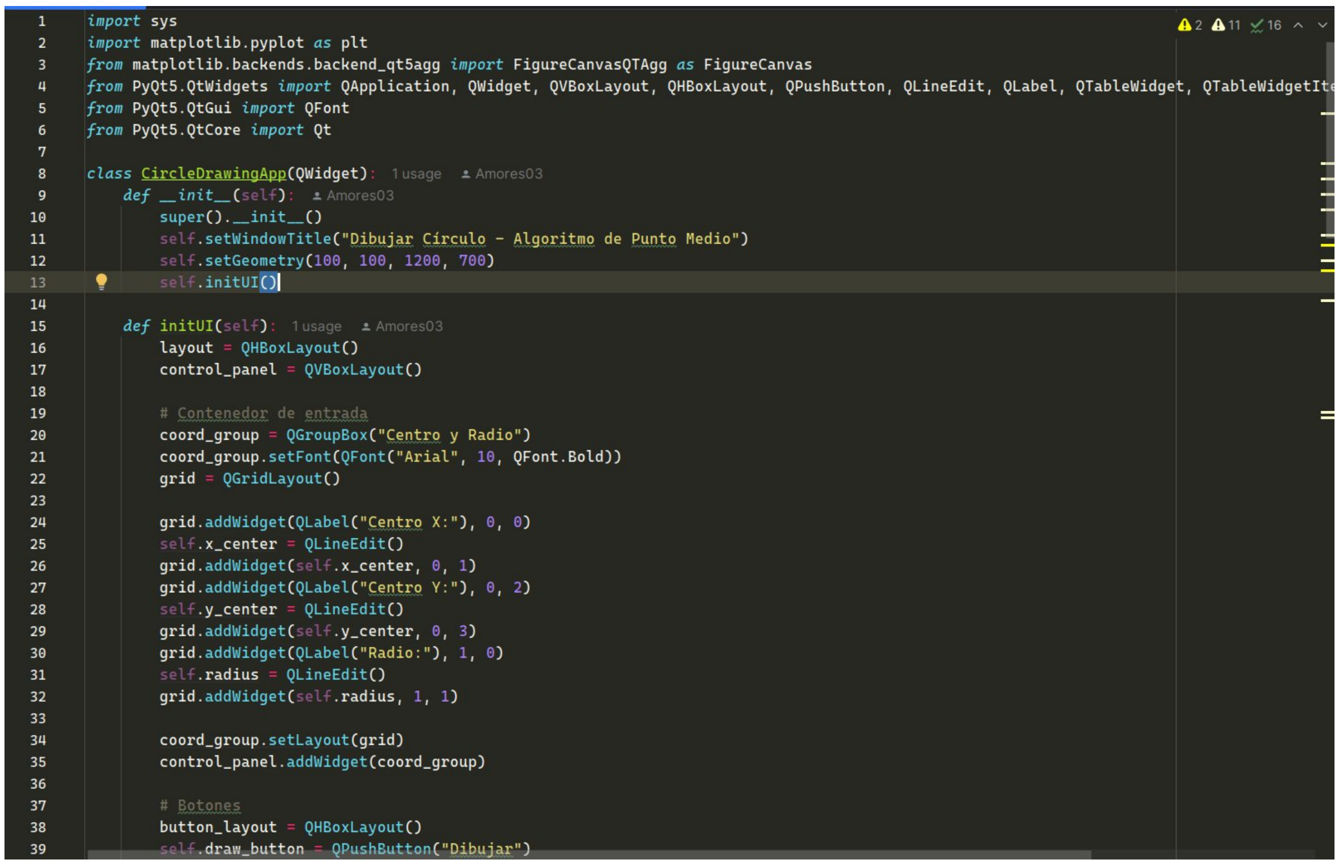
Puntos (-X, Y)

	-X	Y
1	90	150
2	90	149
3	90	148
4	90	147
5	91	146
6	91	145
7	92	144
8	93	143

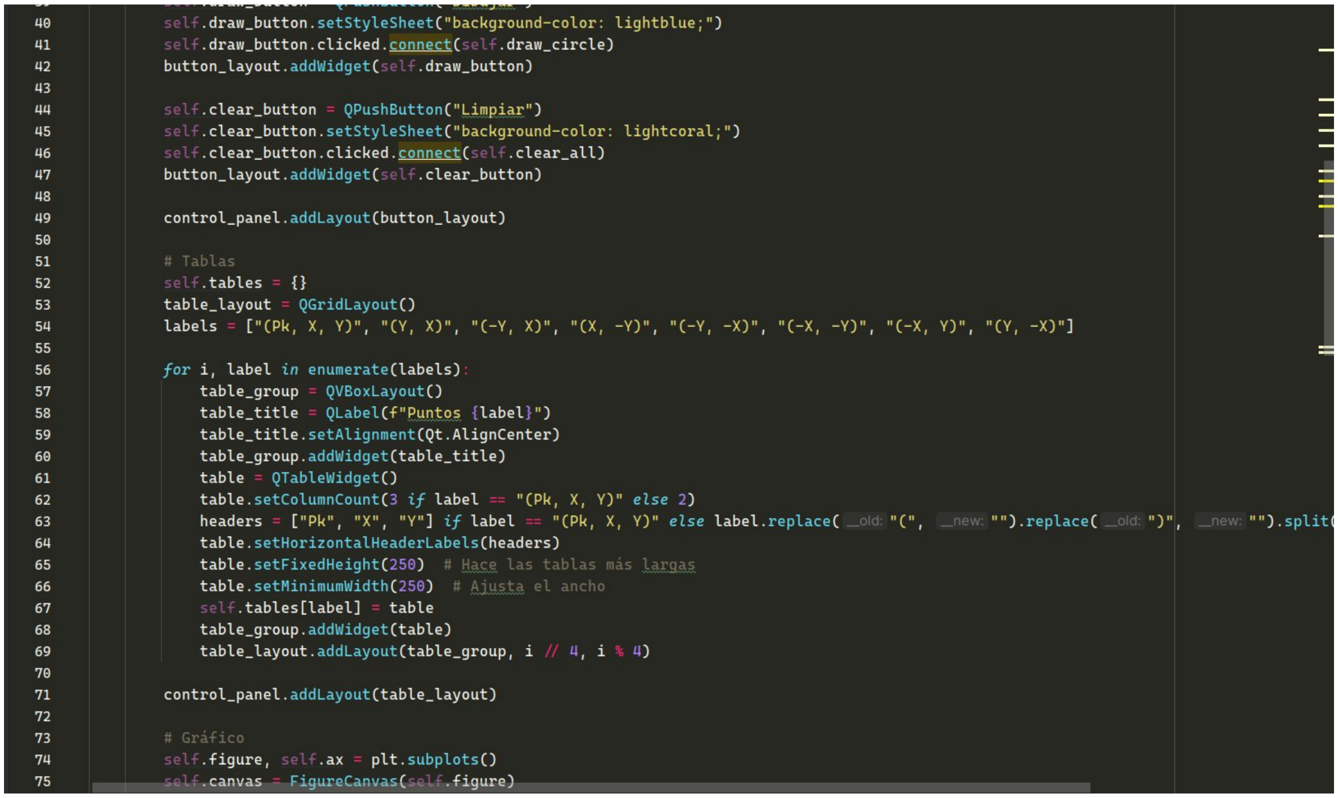
Puntos (Y, -X)

	Y	-X
1	110	150
2	110	149
3	110	148
4	110	147
5	109	146
6	109	145
7	108	144
8	107	143

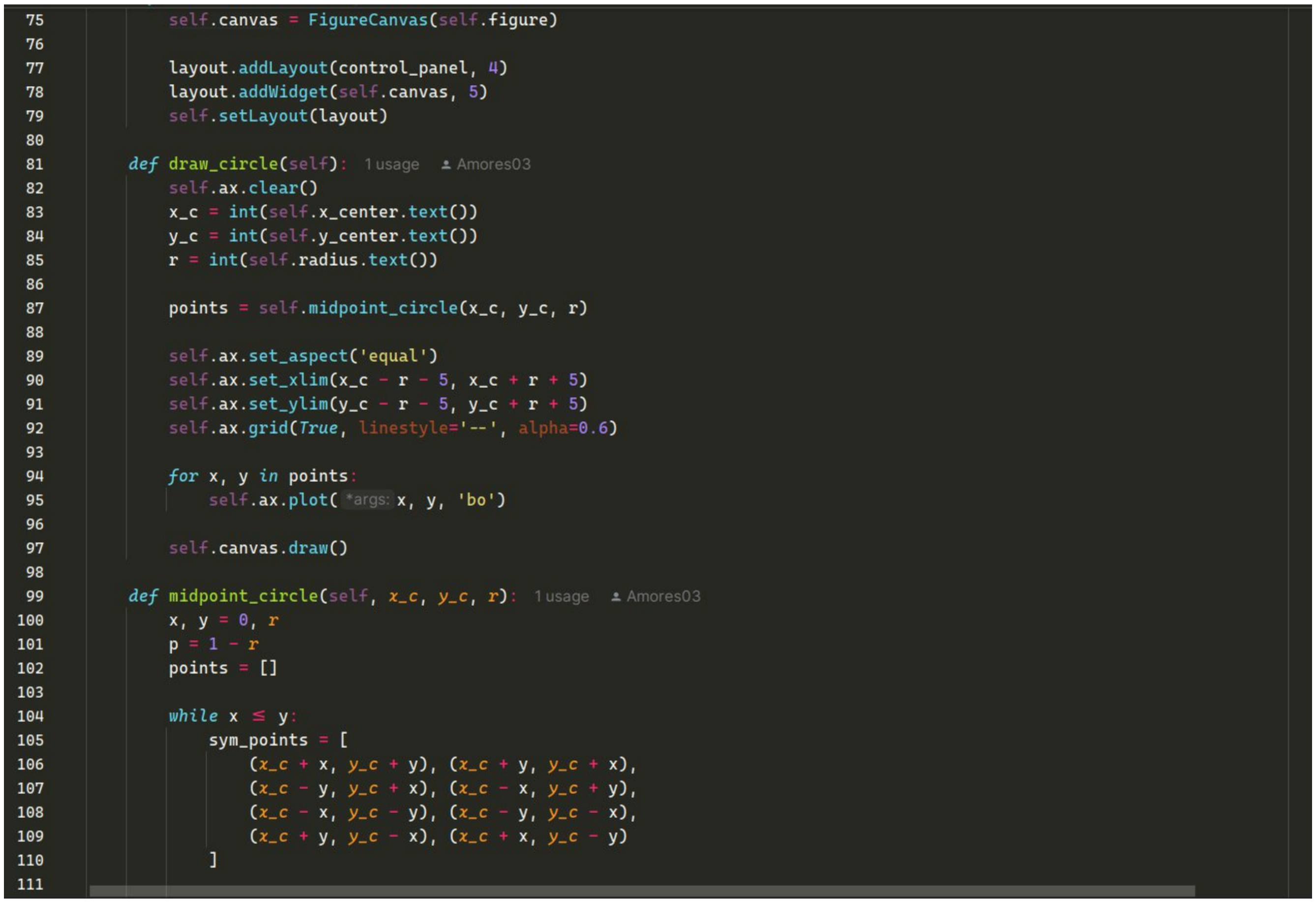




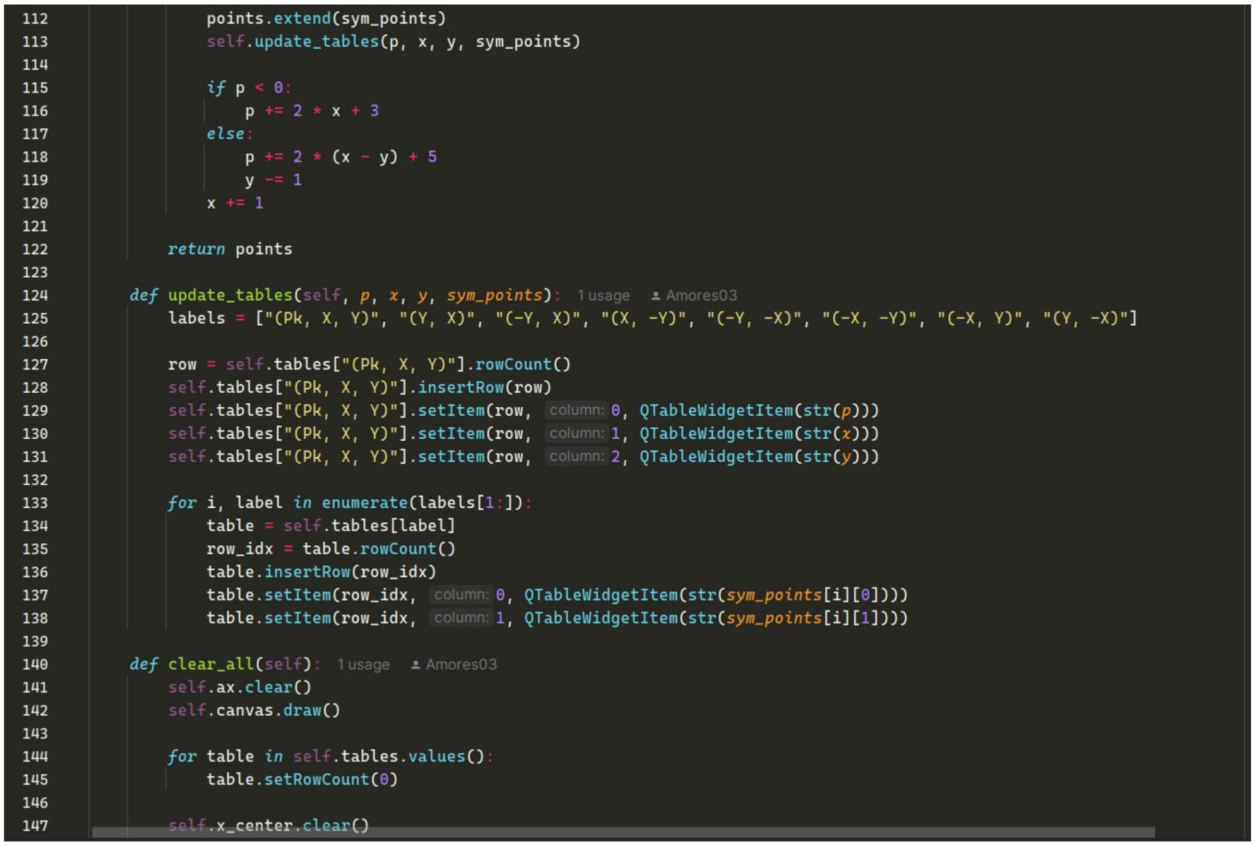
```python
import sys
import matplotlib.pyplot as plt
from matplotlib.backends.backend_qt5agg import FigureCanvasQTAgg as FigureCanvas
from PyQt5.QtWidgets import QApplication, QWidget, QVBoxLayout, QHBoxLayout, QPushButton, QLineEdit, QLabel, QTableWidget, QTableWidgetIte
from PyQt5.QtGui import QFont
from PyQt5.QtCore import Qt

class CircleDrawingApp(QWidget):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Dibujar Círculo - Algoritmo de Punto Medio")
        self.setGeometry(100, 100, 1200, 700)
        self.initUI()

    def initUI(self):
        layout = QHBoxLayout()
        control_panel = QVBoxLayout()

        # Contenedor de entrada
        coord_group = QGroupBox("Centro y Radio")
        coord_group.setFont(QFont("Arial", 10, QFont.Bold))
        grid = QGridLayout()

        grid.addWidget(QLabel("Centro X:"), 0, 0)
        self.x_center = QLineEdit()
        grid.addWidget(self.x_center, 0, 1)
        grid.addWidget(QLabel("Centro Y:"), 0, 2)
        self.y_center = QLineEdit()
        grid.addWidget(self.y_center, 0, 3)
        grid.addWidget(QLabel("Radio:"), 1, 0)
        self.radius = QLineEdit()
        grid.addWidget(self.radius, 1, 1)

        coord_group.setLayout(grid)
        control_panel.addWidget(coord_group)

        # Botones
        button_layout = QHBoxLayout()
        self.draw_button = QPushButton("Dibujar")
```



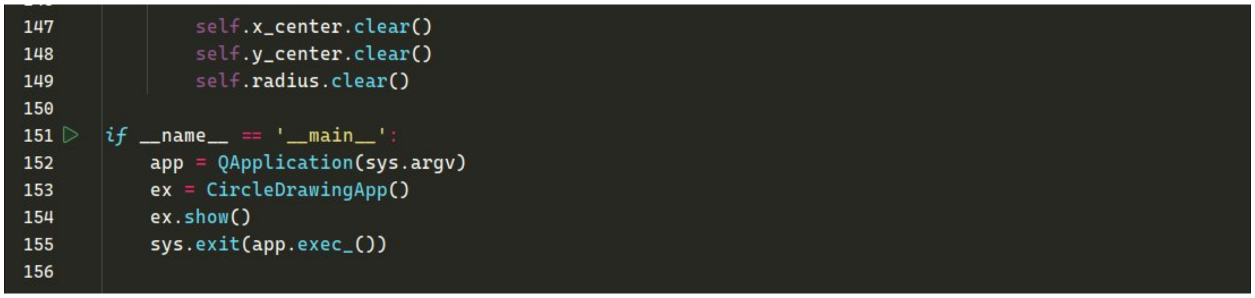
```python
        self.draw_button = QPushButton("Dibujar")
        self.draw_button.setStyleSheet("background-color: lightblue;")
        self.draw_button.clicked.connect(self.draw_circle)
        button_layout.addWidget(self.draw_button)

        self.clear_button = QPushButton("Limpiar")
        self.clear_button.setStyleSheet("background-color: lightcoral;")
        self.clear_button.clicked.connect(self.clear_all)
        button_layout.addWidget(self.clear_button)

        control_panel.addLayout(button_layout)

        # Tablas
        self.tables = {}
        table_layout = QGridLayout()
        labels = ["(Pk, X, Y)", "(Y, X)", "(-Y, X)", "(X, -Y)", "(-Y, -X)", "(-X, -Y)", "(-X, Y)", "(Y, -X)"]

        for i, label in enumerate(labels):
            table_group = QVBoxLayout()
            table_title = QLabel(f"Puntos {label}")
            table_title.setAlignment(Qt.AlignCenter)
            table_group.addWidget(table_title)
            table = QTableWidget()
            table.setColumnCount(3 if label == "(Pk, X, Y)" else 2)
            headers = ["Pk", "X", "Y"] if label == "(Pk, X, Y)" else label.replace( __old: "(",  __new: "").replace( __old: ")",  __new: "").split(
            table.setHorizontalHeaderLabels(headers)
            table.setFixedHeight(250)  # Hace las tablas más largas
            table.setMinimumWidth(250)  # Ajusta el ancho
            self.tables[label] = table
            table_group.addWidget(table)
            table_layout.addLayout(table_group, i // 4, i % 4)

        control_panel.addLayout(table_layout)

        # Gráfico
        self.figure, self.ax = plt.subplots()
        self.canvas = FigureCanvas(self.figure)
```



```python
        self.canvas = FigureCanvas(self.figure)

        layout.addLayout(control_panel, 4)
        layout.addWidget(self.canvas, 5)
        self.setLayout(layout)

    def draw_circle(self):  1 usage  Amores03
        self.ax.clear()
        x_c = int(self.x_center.text())
        y_c = int(self.y_center.text())
        r = int(self.radius.text())

        points = self.midpoint_circle(x_c, y_c, r)

        self.ax.set_aspect('equal')
        self.ax.set_xlim(x_c - r - 5, x_c + r + 5)
        self.ax.set_ylim(y_c - r - 5, y_c + r + 5)
        self.ax.grid(True, linestyle='--', alpha=0.6)

        for x, y in points:
            self.ax.plot( *args: x, y, 'bo')

        self.canvas.draw()

    def midpoint_circle(self, x_c, y_c, r):  1 usage  Amores03
        x, y = 0, r
        p = 1 - r
        points = []

        while x ≤ y:
            sym_points = [
                (x_c + x, y_c + y), (x_c + y, y_c + x),
                (x_c - y, y_c + x), (x_c - x, y_c + y),
                (x_c - x, y_c - y), (x_c - y, y_c - x),
                (x_c + y, y_c - x), (x_c + x, y_c - y)
            ]

```



```python
                points.extend(sym_points)
                self.update_tables(p, x, y, sym_points)

                if p < 0:
                    p += 2 * x + 3
                else:
                    p += 2 * (x - y) + 5
                    y -= 1
                x += 1

            return points

        def update_tables(self, p, x, y, sym_points):  1 usage   Amores03
            labels = ["(Pk, X, Y)", "(Y, X)", "(-Y, X)", "(X, -Y)", "(-Y, -X)", "(-X, -Y)", "(-X, Y)", "(Y, -X)"]

            row = self.tables["(Pk, X, Y)"].rowCount()
            self.tables["(Pk, X, Y)"].insertRow(row)
            self.tables["(Pk, X, Y)"].setItem(row,  column: 0, QTableWidgetItem(str(p)))
            self.tables["(Pk, X, Y)"].setItem(row,  column: 1, QTableWidgetItem(str(x)))
            self.tables["(Pk, X, Y)"].setItem(row,  column: 2, QTableWidgetItem(str(y)))

            for i, label in enumerate(labels[1:]):
                table = self.tables[label]
                row_idx = table.rowCount()
                table.insertRow(row_idx)
                table.setItem(row_idx,  column: 0, QTableWidgetItem(str(sym_points[i][0])))
                table.setItem(row_idx,  column: 1, QTableWidgetItem(str(sym_points[i][1])))

        def clear_all(self):  1 usage   Amores03
            self.ax.clear()
            self.canvas.draw()

            for table in self.tables.values():
                table.setRowCount(0)

            self.x_center.clear()
```



```python
            self.x_center.clear()
            self.y_center.clear()
            self.radius.clear()

    if __name__ == '__main__':
        app = QApplication(sys.argv)
        ex = CircleDrawingApp()
        ex.show()
        sys.exit(app.exec_())
```

Tarea	Responsable	Día 1	Día 2	Día 3	Día 4	Día 5	Día 6	Día 7
Planificación y análisis	Todos	🟦						
Diseño de interfaz	Daniel		🟦	🟦				
Implementación del algoritmo DDA	Cristóbal			🟦	🟦			
Integración con la interfaz	Adrián				🟦	🟦		
Pruebas y corrección de errores	Cristobal					🟦	🟦	
Optimización del código	Adrián					🟦	🟦	
Documentación y entrega	Daniel						🟦	🟦