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.

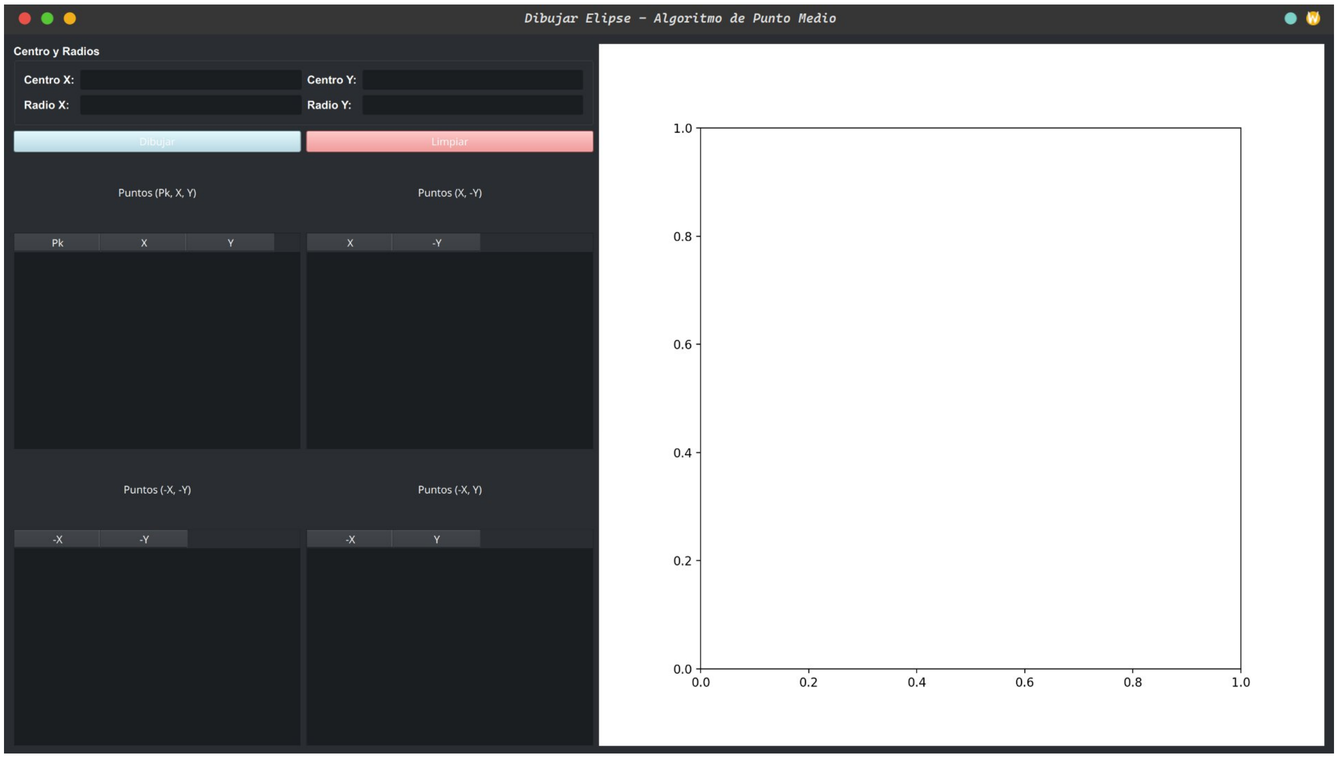
ALUMNOS:

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

“4to. DOCUMENTO PROGRAMA ELIPSE RELLENA”

FECHA DE ENTREGA: 12 DE ABRIL DEL 2025.

CAPTURA DE PANTALLA DEL PROGRAMA



Centro y Radios

Centro X:
Centro Y:
Radio X:
Radio Y:

Puntos (Pk, X, Y)

Pk	X	Y

Puntos (X, -Y)

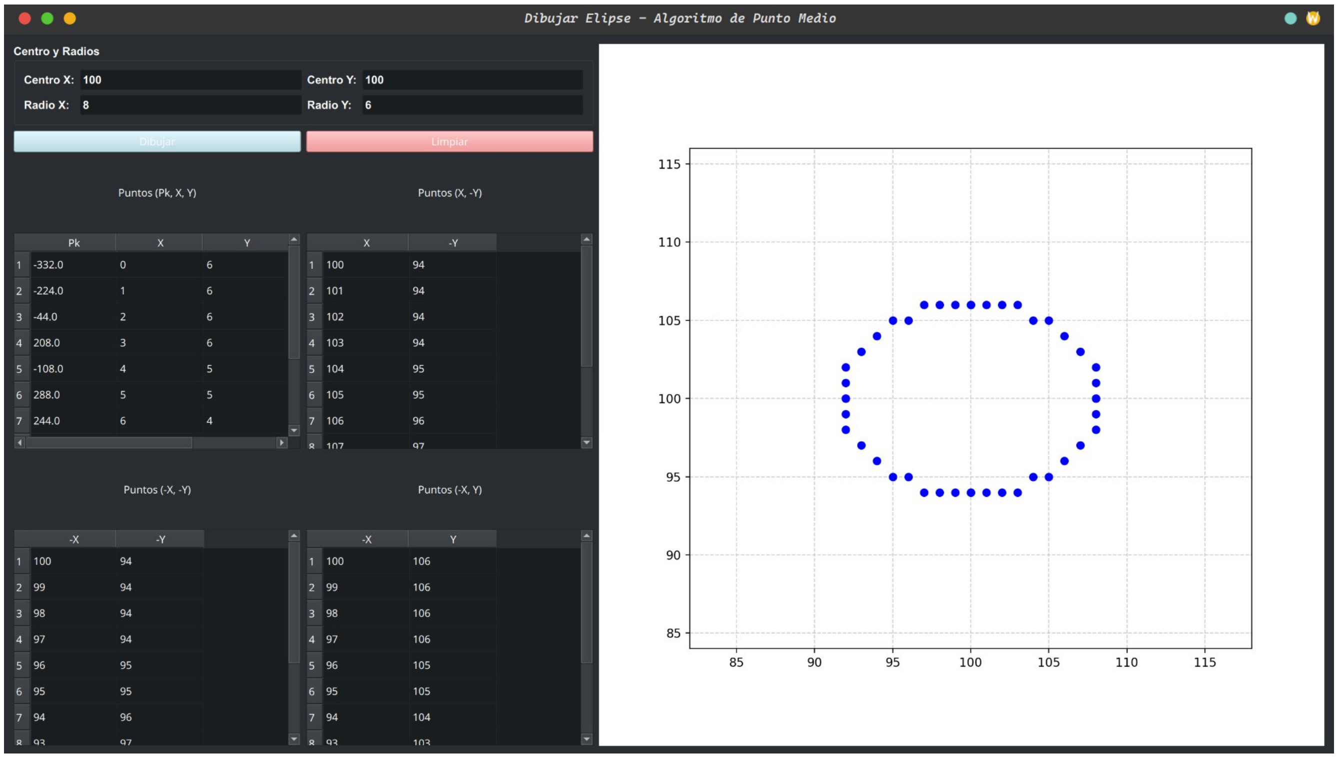
X	-Y

Puntos (-X, -Y)

-X	-Y

Puntos (-X, Y)

-X	Y



Centro y Radios

Centro X: 100
Centro Y: 100
Radio X: 8
Radio Y: 6

Puntos (Pk, X, Y)

	Pk	X	Y
1	-332.0	0	6
2	-224.0	1	6
3	-44.0	2	6
4	208.0	3	6
5	-108.0	4	5
6	288.0	5	5
7	244.0	6	4

Puntos (X, -Y)

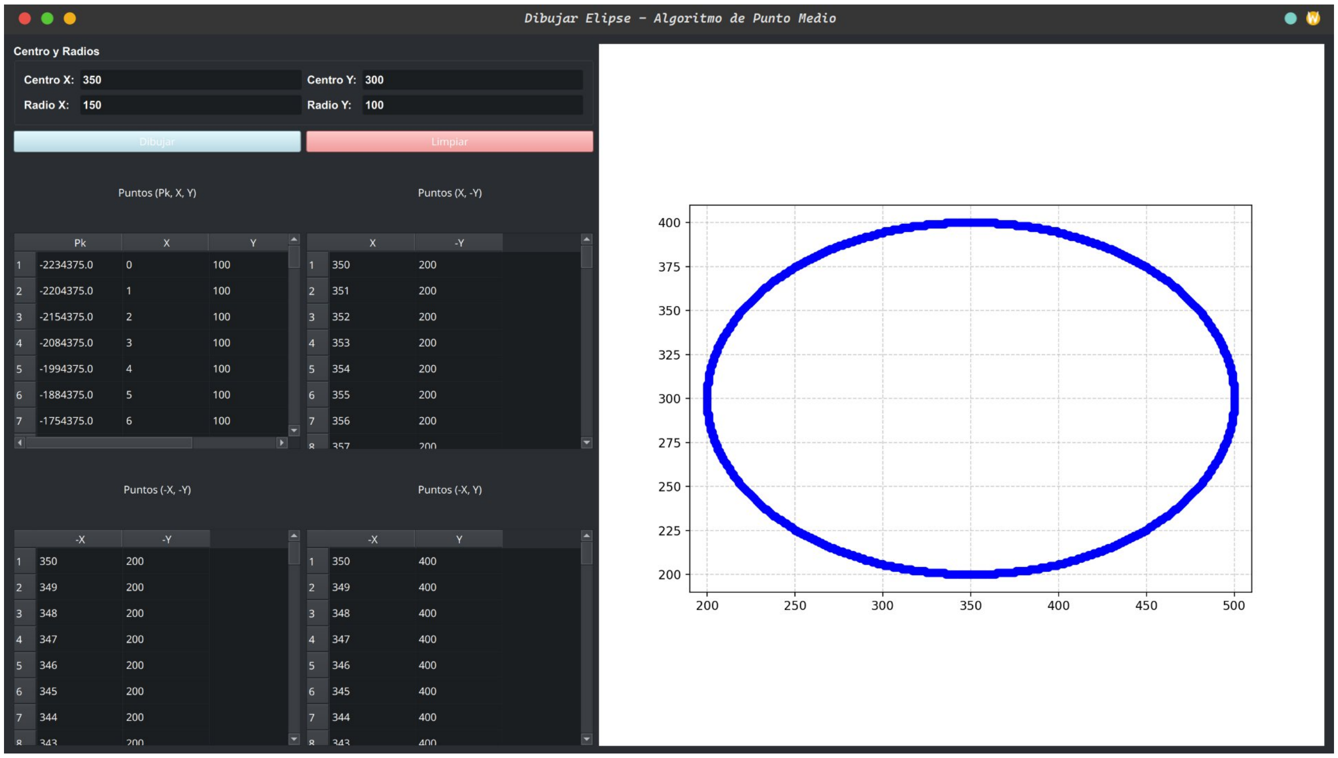
	X	-Y
1	100	94
2	101	94
3	102	94
4	103	94
5	104	95
6	105	95
7	106	96
8	107	97

Puntos (-X, -Y)

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

Puntos (-X, Y)

	-X	Y
1	100	106
2	99	106
3	98	106
4	97	106
5	96	105
6	95	105
7	94	104
8	93	103



Centro y Radios

Centro X: 350 Centro Y: 300

Radio X: 150 Radio Y: 100

Puntos (Pk, X, Y)

	Pk	X	Y
1	-2234375.0	0	100
2	-2204375.0	1	100
3	-2154375.0	2	100
4	-2084375.0	3	100
5	-1994375.0	4	100
6	-1884375.0	5	100
7	-1754375.0	6	100

Puntos (X, -Y)

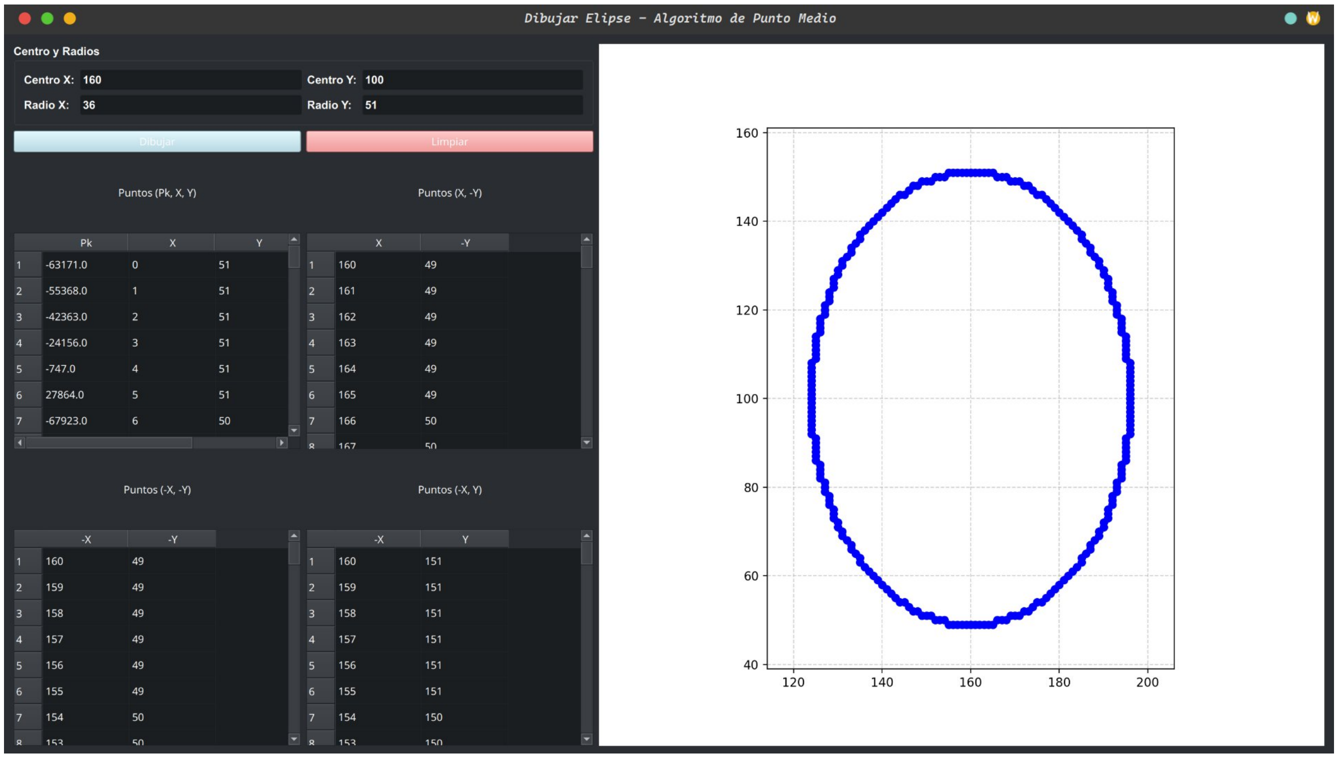
	X	-Y
1	350	200
2	351	200
3	352	200
4	353	200
5	354	200
6	355	200
7	356	200
8	357	200

Puntos (-X, -Y)

	-X	-Y
1	350	200
2	349	200
3	348	200
4	347	200
5	346	200
6	345	200
7	344	200
8	343	200

Puntos (-X, Y)

	-X	Y
1	350	400
2	349	400
3	348	400
4	347	400
5	346	400
6	345	400
7	344	400
8	343	400



Centro y Radios

Centro X: 160 Centro Y: 100

Radio X: 36 Radio Y: 51

Puntos (Pk, X, Y)

	Pk	X	Y
1	-63171.0	0	51
2	-55368.0	1	51
3	-42363.0	2	51
4	-24156.0	3	51
5	-747.0	4	51
6	27864.0	5	51
7	-67923.0	6	50

Puntos (X, -Y)

	X	-Y
1	160	49
2	161	49
3	162	49
4	163	49
5	164	49
6	165	49
7	166	50
8	167	50

Puntos (-X, -Y)

	-X	-Y
1	160	49
2	159	49
3	158	49
4	157	49
5	156	49
6	155	49
7	154	50
8	153	50

Puntos (-X, Y)

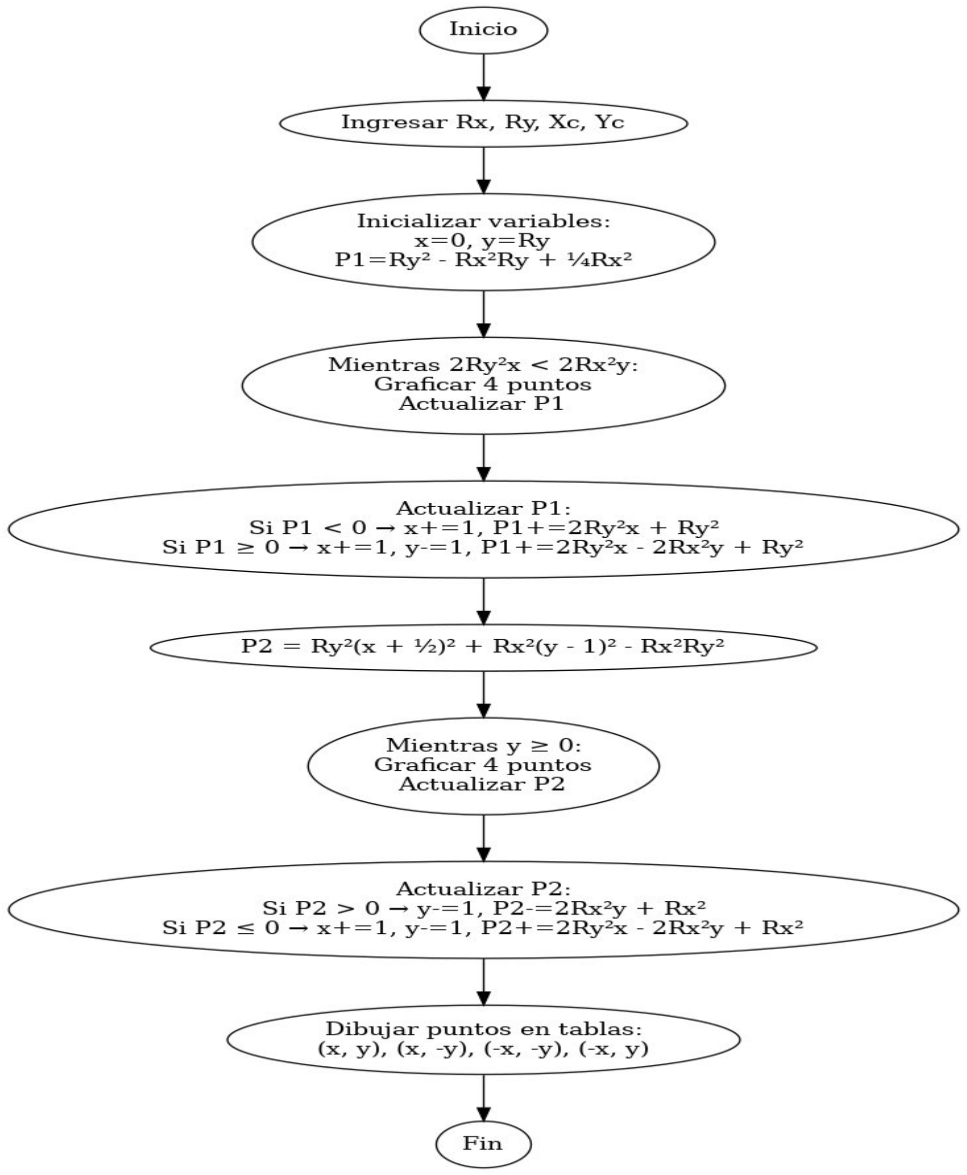
	-X	Y
1	160	151
2	159	151
3	158	151
4	157	151
5	156	151
6	155	151
7	154	150
8	153	150

CAPTURA DE CODIGO FUENTE

```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, QTableWidgetItem, QGroupBox, QGridLayout
from PyQt5.QtGui import QFont
from PyQt5.QtCore import Qt


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

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

        # Entradas
        input_group = QGroupBox("Centro y Radios")
        input_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 X:"), 1, 0)
        self.rx = QLineEdit()
        grid.addWidget(self.rx, 1, 1)
        grid.addWidget(QLabel("Radio Y:"), 1, 2)
        self.ry = QLineEdit()
        grid.addWidget(self.ry, 1, 3)

        input_group.setLayout(grid)
        control_panel.addWidget(input_group)

        # Botones
        button_layout = QHBoxLayout()
        self.draw_button = QPushButton("Dibujar")
        self.draw_button.setStyleSheet("background-color: lightblue;")
        self.draw_button.clicked.connect(self.draw_ellipse)
        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 = {}
        labels = ["(Pk, X, Y)", "(X, -Y)", "(-X, -Y)", "(-X, Y)"]
        table_layout = QGridLayout()

        for i, label in enumerate(labels):
            group = QVBoxLayout()
            title = QLabel(f"Puntos {label}")
            title.setAlignment(Qt.AlignCenter)
            group.addWidget(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("(",
"").replace(")", "").split(", ")
            table.setHorizontalHeaderLabels(headers)
            table.setFixedHeight(250)
            table.setMinimumWidth(250)
            self.tables[label] = table
            group.addWidget(table)
            table_layout.addLayout(group, i // 2, i % 2)

        control_panel.addLayout(table_layout)

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

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

    def draw_ellipse(self):
        self.ax.clear()
        x_c = int(self.x_center.text())
        y_c = int(self.y_center.text())
        rx = int(self.rx.text())
        ry = int(self.ry.text())

        points = self.midpoint_ellipse(x_c, y_c, rx, ry)

        self.ax.set_aspect('equal')
        self.ax.set_xlim(x_c - rx - 10, x_c + rx + 10)
        self.ax.set_ylim(y_c - ry - 10, y_c + ry + 10)
        self.ax.grid(True, linestyle='--', alpha=0.6)

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

        self.canvas.draw()

    def midpoint_ellipse(self, xc, yc, rx, ry):
        x = 0
        y = ry
        rx2 = rx * rx
        ry2 = ry * ry
        tworx2 = 2 * rx2
        twory2 = 2 * ry2
        px = 0
        py = tworx2 * y
        points = []

        # Region 1
        p1 = ry2 - (rx2 * ry) + (0.25 * rx2)
        while px < py:
            sym_points = self.plot_symmetry(x, y, xc, yc)
            points.extend(sym_points)
            self.update_tables(p1, x, y, sym_points)

            x += 1
            px += twory2
            if p1 < 0:
                p1 += ry2 + px
            else:
                y -= 1
                py -= tworx2
                p1 += ry2 + px - py
```

```python
        # Region 2
        p2 = ry2 * (x + 0.5) ** 2 + rx2 * (y - 1) ** 2 - rx2 * ry2
        while y >= 0:
            sym_points = self.plot_symmetry(x, y, xc, yc)
            points.extend(sym_points)
            self.update_tables(p2, x, y, sym_points)

            y -= 1
            py -= tworx2
            if p2 > 0:
                p2 += rx2 - py
            else:
                x += 1
                px += twory2
                p2 += rx2 - py + px

        return points

    def plot_symmetry(self, x, y, xc, yc):
        return [
            (xc + x, yc + y),  # original
            (xc + x, yc - y),
            (xc - x, yc - y),
            (xc - x, yc + y),
        ]

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

        labels = ["(X, -Y)", "(-X, -Y)", "(-X, Y)"]
        for i, label in enumerate(labels):
            table = self.tables[label]
            r = table.rowCount()
            table.insertRow(r)
            table.setItem(r, 0, QTableWidgetItem(str(sym_points[i + 1][0])))
            table.setItem(r, 1, QTableWidgetItem(str(sym_points[i + 1][1])))

    def clear_all(self):
        self.ax.clear()
        self.canvas.draw()
        for table in self.tables.values():
            table.setRowCount(0)
        self.x_center.clear()
        self.y_center.clear()
        self.rx.clear()
        self.ry.clear()


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

DIAGRAMA GANT Y FLUJO


Inicio
Ingresar Rx, Ry, Xc, Yc
Inicializar variables:
x=0, y=Ry
$P1=Ry^2 - Rx^2Ry + \frac{1}{4}Rx^2$
Mientras $2Ry^2x < 2Rx^2y$:
Graficar 4 puntos
Actualizar P1
Actualizar P1:
Si $P1 < 0 \rightarrow x+=1, P1+=2Ry^2x + Ry^2$
Si $P1 \geq 0 \rightarrow x+=1, y-=1, P1+=2Ry^2x - 2Rx^2y + Ry^2$
$P2 = Ry^2(x + \frac{1}{2})^2 + Rx^2(y - 1)^2 - Rx^2Ry^2$
Mientras $y \geq 0$:
Graficar 4 puntos
Actualizar P2
Actualizar P2:
Si $P2 > 0 \rightarrow y-=1, P2-=2Rx^2y + Rx^2$
Si $P2 \leq 0 \rightarrow x+=1, y-=1, P2+=2Ry^2x - 2Rx^2y + Rx^2$
Dibujar puntos en tablas:
(x, y), (x, -y), (-x, -y), (-x, y)
Fin

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	Cristóbal			🟦	🟦			
Integración con la interfaz	Adrián				🟦	🟦		
Pruebas y corrección de errores	Cristobal					🟦	🟦	
Optimización del código	Adrián					🟦	🟦	
Documentación y entrega	Daniel						🟦	🟦