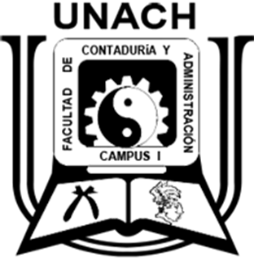
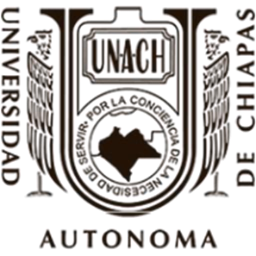
UNIVERSIDAD AUTÓNOMA DE CHIAPAS.



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.

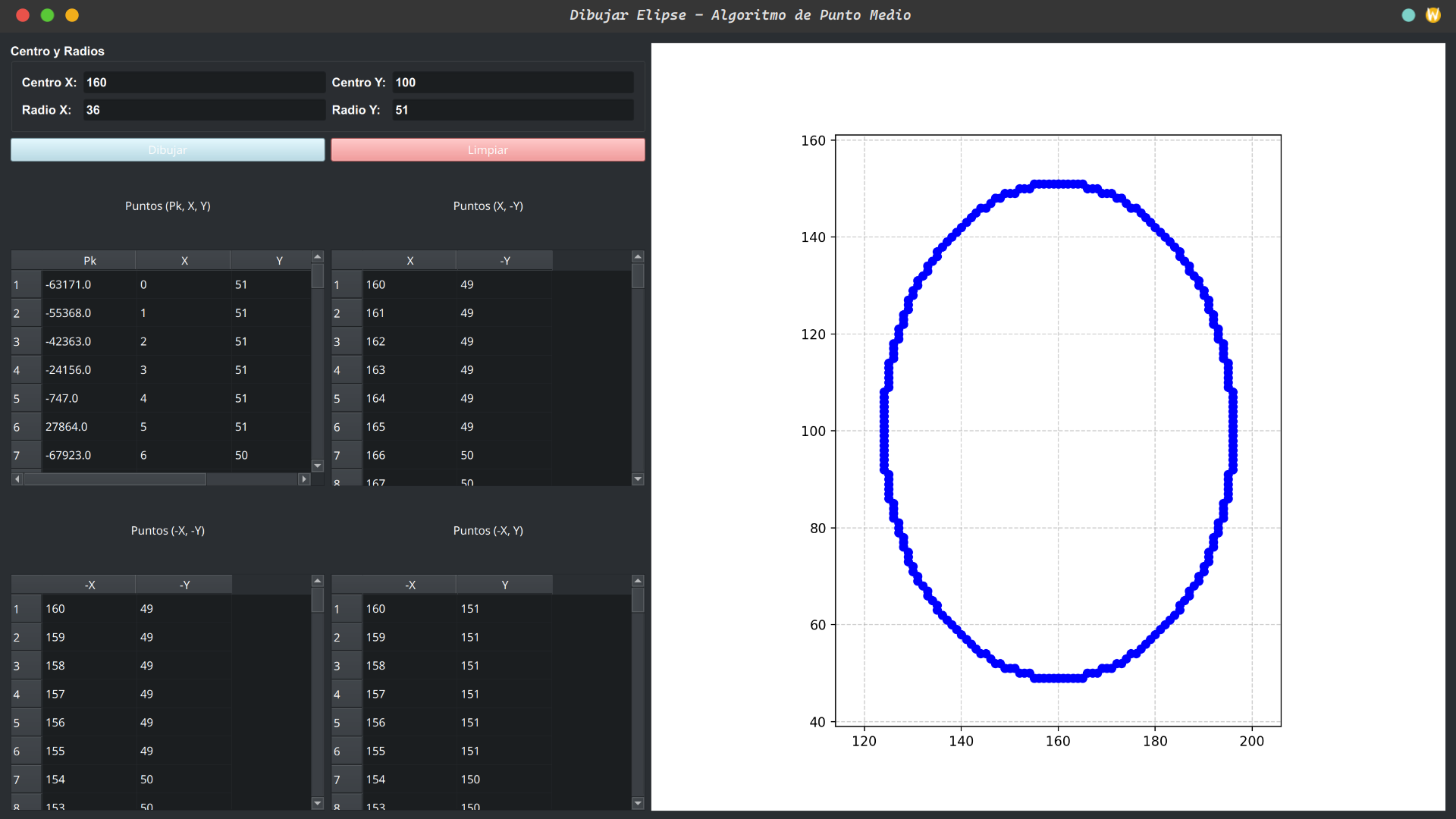
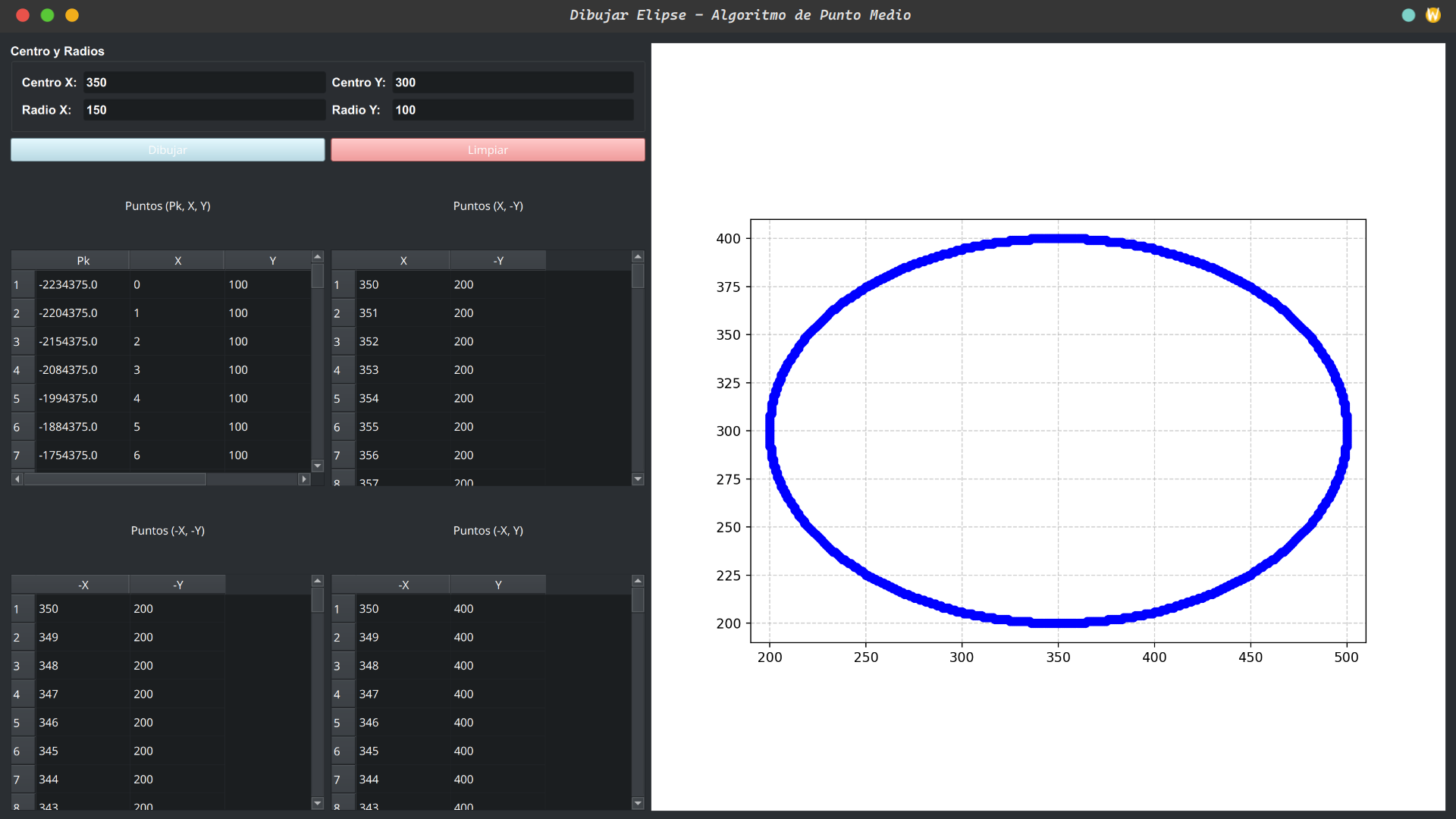
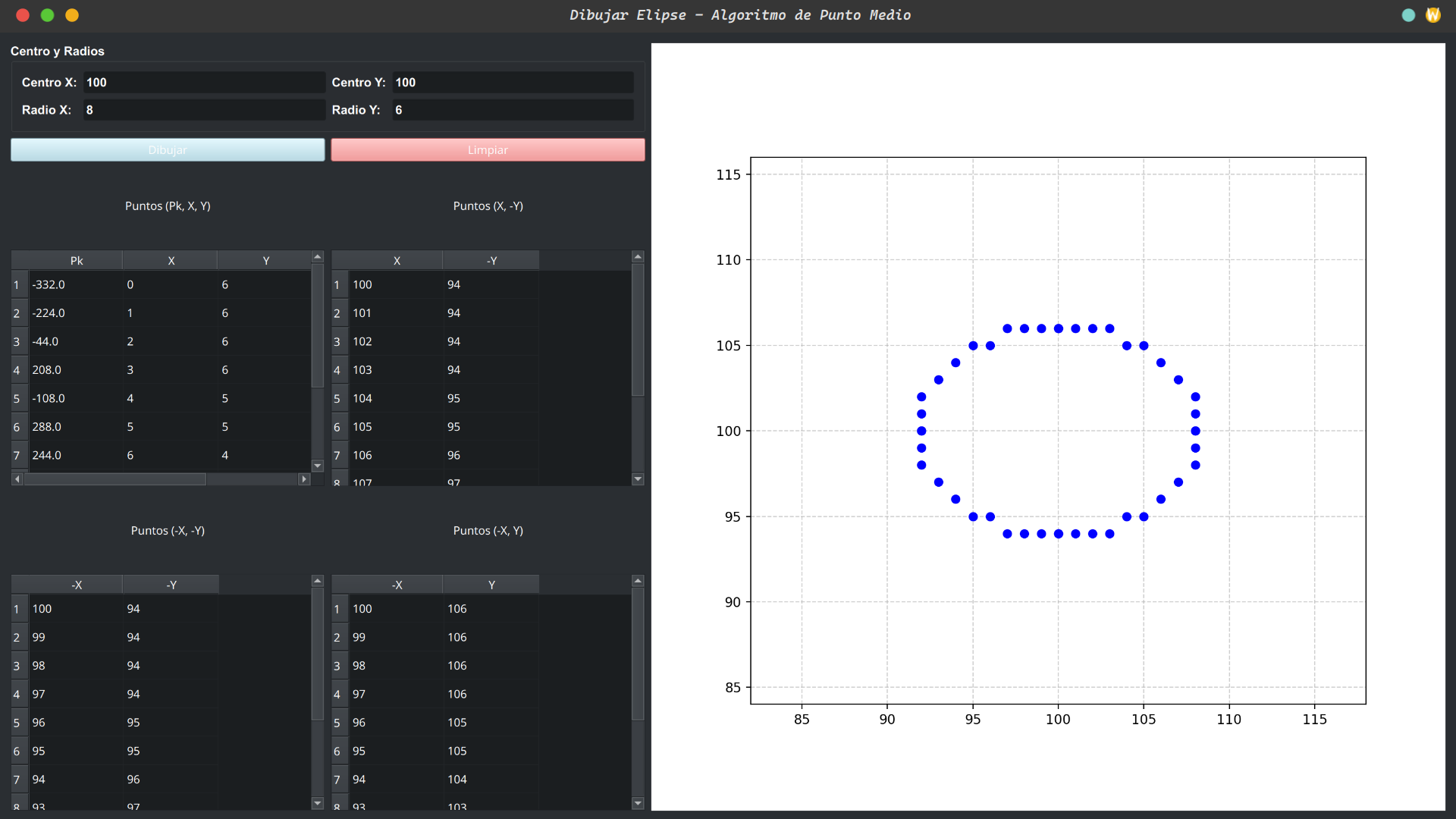
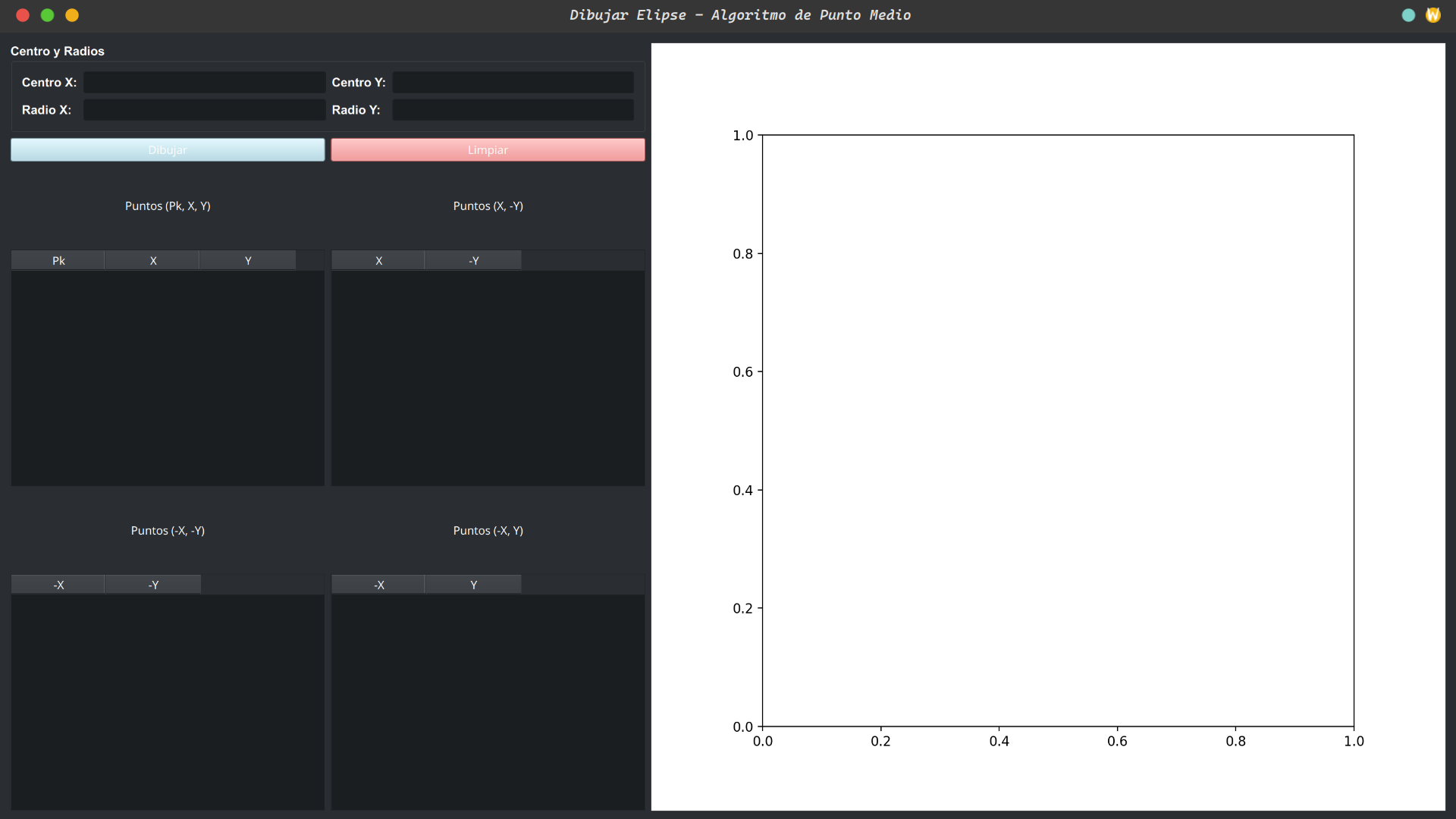
ALUMNOS:

* 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.

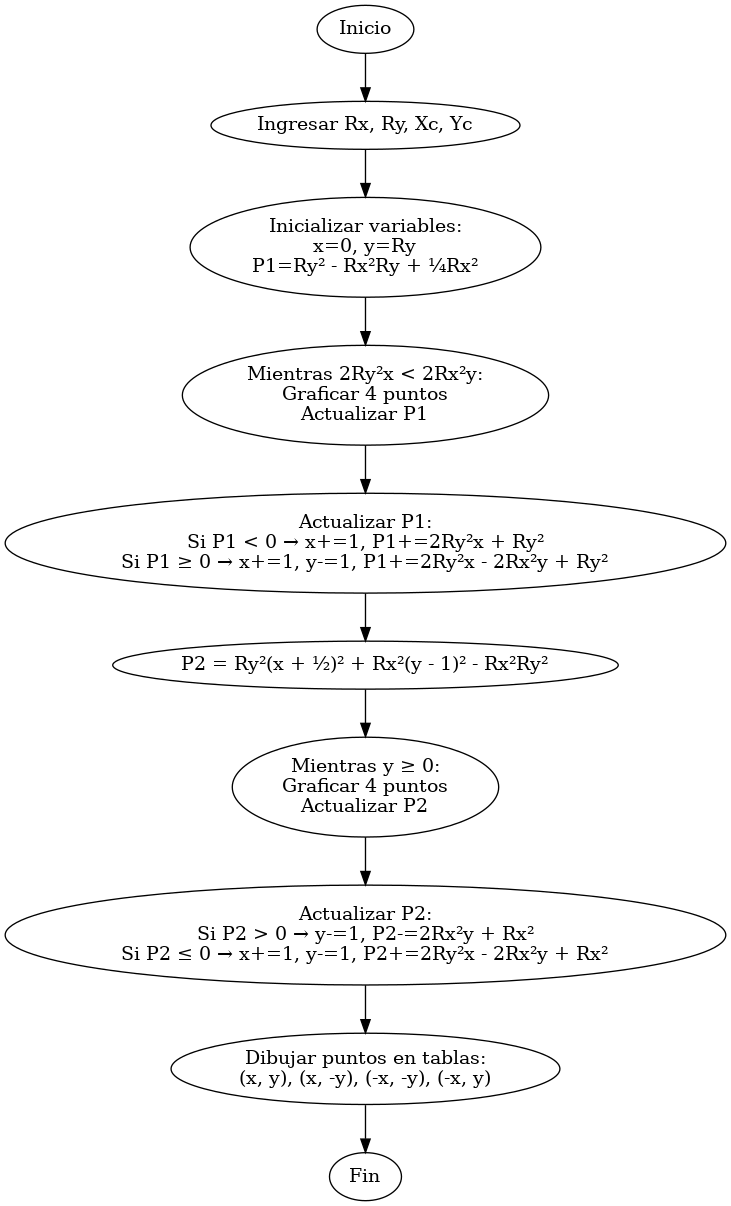
CAPTURA DE PANTALLA DEL PROGRAMA



CAPTURA DE CODIGO FUENTE

*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  
  
 # 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



| 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 |  |  |  |  |  |  |  |