Sixth Part

باید بدانیم که PyQt6 از کلاس `QPainter` برای تمام عملیات نقاشی (bitmap graphics) استفاده می‌کند. کد نقاشی باید داخل متد `paintEvent()` ویجت قرار گیرد؛ این متد هنگام نیاز به بازطراحی ویجت (مثلاً resize یا explicit update) فراخوانی می‌شود. با `QPen` خطوط، با `QBrush` پرکننده‌ها و با `QPixmap` تصاویر را رسم می‌کنیم. همچنین می‌توان ویجت خود را با ارث‌بری از `QWidget` توسعه داد و درون `paintEvent` هر چیزی از اشکال ساده تا مسیرهای پیچیده را رسم کرد. در تمرین نهایی، یک بوم نقاشی پیاده خواهیم کرد که با حرکت ماوس روی آن خطوط ترسیم می‌کند.

۱. آغاز QPainter و چرخه‌ی نقاشی

۱.۱ خب QPainter چیست؟

- `QPainter` رابط low-level برای تمام عملیات bitmap graphics در Qt است؛ از رسم خطوط و مستطیل تا تصاویر و متن Python GUIs.
- تمام عملیات گرافیکی باید بین متدهای `begin()` و `end()` یا درون `paintEvent()` انجام شوند ZetCode.

۱.۲ چرخه‌ی نقاشی (Paint Cycle)

۱. **رویداد paintEvent**: زمانی که ویجت نیاز به redraw دارد (اولین نمایش، resize، `update()`، `repaint()`)، Qt متد `paintEvent(self, event)` را فراخوانی می‌کند ZetCode.

۲. **ایجاد QPainter**: داخل `paintEvent` یک شیء `QPainter(self)` بسازید.

۳. **اعمال عملیات گرافیکی**: با متدهایی مثل `drawLine()`, `drawRect()`, `drawPixmap()`, `drawText()` عملیات را اجرا کنید.

۴. **اتمام نقاشی**: هنگام خروج از متد `paintEvent`، QPainter به‌صورت خودکار پایان می‌یابد.

۲. پیاده‌سازی paintEvent

۲.۱ مثال رسم خط و مستطیل

```
import sys
from PyQt6.QtWidgets import QWidget, QApplication
from PyQt6.QtGui import QPainter, QPen
from PyQt6.QtCore import Qt

class ShapeWidget(QWidget):
    def paintEvent(self, event):
        painter = QPainter(self)                    # create painter :cc
        pen = QPen()                                # default pen
        pen.setWidth(3)                             # line width :conter
        painter.setPen(pen)                         # assign pen to pair
        painter.drawLine(10, 10, 100, 10)           # draw a horizontal
        painter.drawRect(10, 20, 80, 50)            # draw rectangle :cc

if __name__ == '__main__':
    app = QApplication(sys.argv)
    w = ShapeWidget()
    w.resize(200, 100)
    w.show()
    sys.exit(app.exec_())
```

۲.۲ رسم دایره و بیضی با QPainter

```
    painter.drawEllipse(120, 20, 60, 40)            # draw ellipse :conte
```

۳. استفاده از QBrush و رنگ‌آمیزی

- `QBrush` برای پر کردن داخل اشکال با رنگ، الگو یا گرادیانت استفاده می‌شود Like Geeks.
- `painter.setBrush(brush)` پرکننده را تعیین می‌کند.

```
from PyQt6.QtGui import QBrush

brush = QBrush()
```

```
brush.setStyle(Qt.SolidPattern)                    # solid fill :contentF
painter.setBrush(brush)
painter.drawRect(10, 80, 80, 40)                   # filled rectangle
```

۴. رسم تصویر با QPixmap

- `QPixmap` برای بارگذاری و نمایش تصویر به کار می‌رود.
- متد `drawPixmap(x, y, pixmap)` تصویر را در مختصات مشخص رسم می‌کند Python GUIs.

```
from PyQt6.QtGui import QPixmap

pix = QPixmap('logo.png')
painter.drawPixmap(120, 80, pix)                   # draw image at (120,8
```

۵. ساخت ویجت نقاشی تعاملی (Paint App)

در این مثال یک ویجت می‌سازیم که با کلیک و حرکت ماوس روی آن، خطوط رسم کند (simple paint app) w3resource.

```
import sys
from PyQt6.QtWidgets import QWidget, QApplication
from PyQt6.QtGui import QPainter, QPen
from PyQt6.QtCore import Qt, QPoint

class PaintWidget(QWidget):
    def __init__(self):
        super().__init__()
        self.last_point = QPoint()
        self.setWindowTitle('Paint App')
        self.image = QPixmap(self.size())
        self.image.fill(Qt.white)

    def paintEvent(self, event):
        canvas_painter = QPainter(self)
        canvas_painter.drawPixmap(0, 0, self.image)
```

```
    def mousePressEvent(self, event):
        if event.button() == Qt.LeftButton:
            self.last_point = event.pos()

    def mouseMoveEvent(self, event):
        if event.buttons() & Qt.LeftButton:
            painter = QPainter(self.image)
            pen = QPen()
            pen.setWidth(5)
            painter.setPen(pen)
            painter.drawLine(self.last_point, event.pos())
            self.last_point = event.pos()
            self.update()                              # trigger paintEvent

if __name__ == '__main__':
    app = QApplication(sys.argv)
    w = PaintWidget()
    w.resize(400, 300)
    w.show()
    sys.exit(app.exec())
```

۶. نکات پیشرفته

- برای رسم مسیرهای پیچیده از `QPainterPath` استفاده کنید Stack Overflow.
- بهینه‌سازی: در صورت رسم مکرر، از buffer (QPixmap) برای کاهش flicker بهره ببرید (همانند مثال بالا) w3resource.
- می‌توان با Qt Style Sheets رنگ‌ها و قلم‌ها را از طریق فایل QSS تنظیم کرد و بدون تغییر کد، ظاهر را تغییر داد Qt Wiki.

تمرین‌های پیشنهادی

۱. رسم چندضلعی (polygon) با استفاده از `drawPolygon()` و لیستی از نقاط.

۲. ساخت ابزار خط‌کش: هنگام حرکت ماوس، مختصات نقطه جاری را در گوشه‌ی پنجره نمایش بده.

۳. رسم گرادیانت خطی یا شعاعی با `QLinearGradient` یا `QRadialGradient`.

پس از انجام این تمرین‌ها، درک کاملی از گرافیک دوبعدی در PyQt5 خواهی داشت و آماده‌ای برای **فصل ۹: پروژه‌های واقعی**.