QPixmap・QIcon リファレンス

QPixmapとQIconは、PySide6における画像とアイコンの処理を担当する重要なクラスです。

QPixmap

概要

QPixmapは画像データを保持し、表示に最適化されたクラスです。メモリ効率が良く、高速な描画が可能です。

基本的な作成方法

```
# ファイルから読み込み
pixmap = QPixmap("image.png")

# サイズを指定して作成
pixmap = QPixmap(100, 100)

# 空のピクセルマップ
pixmap = QPixmap()
```

よく使用されるメソッド

サイズ関連

- `width()` - 幅を取得
- `height()` - 高さを取得
- `size()` - QSizeとしてサイズを取得
- `scaled(width, height, mode)` - スケールした複製を作成

変形・変換

- `scaled()` - サイズ変更
- `transformed()` - 回転・反転等の変形
- `copy(x, y, width, height)` - 部分的にコピー

保存・読み込み

- `load(filename)` - ファイル読み込み
- `save(filename, format)` - ファイル保存
- `loadFromData(data)` - バイトデータから読み込み

状態確認

- `isNull()` - 空かどうかチェック
- `width()`, `height()` - サイズ取得

スケーリングモード

```
from PySide6.QtCore import Qt

# アスペクト比を保持して拡大縮小
pixmap.scaled(100, 100, Qt.KeepAspectRatio)

# アスペクト比を無視して拡大縮小
pixmap.scaled(100, 100, Qt.IgnoreAspectRatio)

# アスペクト比を保持し、はみ出る部分をクロップ
pixmap.scaled(100, 100, Qt.KeepAspectRatioByExpanding)
```

変形の種類

```
from PySide6.QtGui import QTransform

# 回転
transform = QTransform()
transform.rotate(45)
rotated_pixmap = pixmap.transformed(transform)

# 反転
```

```
flipped_h = pixmap.transformed(QTransform().scale(-1, 1))  # 水平反転
flipped_v = pixmap.transformed(QTransform().scale(1, -1))  # 垂直反転
```

QIcon

概要

QIconは複数のサイズ・状態のピクセルマップを管理し、UI要素で使用されるアイコンを表現します。

基本的な作成方法

```
# ファイルから作成
icon = QIcon("icon.png")

# QPixmapから作成
icon = QIcon(pixmap)

# 空のアイコン
icon = QIcon()
```

モードと状態

```
from PySide6.QtGui import QIcon

# モード
QIcon.Normal    # 通常状態
QIcon.Disabled  # 無効状態
QIcon.Active    # アクティブ状態（ホバー等）
QIcon.Selected  # 選択状態

# 状態
QIcon.Off       # オフ状態
QIcon.On        # オン状態
```

複数サイズの管理

```
icon = QIcon()

# 異なるサイズを追加
icon.addFile("icon_16.png", QSize(16, 16))
icon.addFile("icon_32.png", QSize(32, 32))
icon.addFile("icon_64.png", QSize(64, 64))

# QPixmapを直接追加
icon.addPixmap(pixmap_16, QIcon.Normal, QIcon.Off)
icon.addPixmap(pixmap_32, QIcon.Normal, QIcon.Off)
```

よく使用されるメソッド

サイズ関連

- `availableSizes()` - 利用可能なサイズ一覧
- `actualSize(size)` - 指定サイズに最も近い実際のサイズ

ピクセルマップ取得

- `pixmap(size)` - 指定サイズのQPixmapを取得
- `pixmap(width, height)` - 指定サイズのQPixmapを取得

状態確認

- `isNull()` - 空かどうかチェック

実用的な使用例

画像の動的生成

```
# グラデーション画像の生成
pixmap = QPixmap(200, 100)
painter = QPainter(pixmap)
gradient = QLinearGradient(0, 0, 200, 0)
gradient.setColorAt(0, QColor(255, 0, 0))
gradient.setColorAt(1, QColor(0, 0, 255))
painter.fillRect(pixmap.rect(), gradient)
painter.end()
```

アイコンの状態管理

```
# ボタンの状態に応じたアイコン
button_icon = QIcon()
button_icon.addFile("play.png", QSize(), QIcon.Normal, QIcon.Off)
button_icon.addFile("pause.png", QSize(), QIcon.Normal, QIcon.On)
button.setIcon(button_icon)
button.setCheckable(True)  # トグル可能にする
```

画像効果の適用

```
# セピア効果
def apply_sepia(pixmap):
    image = pixmap.toImage()
    for y in range(image.height()):
        for x in range(image.width()):
            color = QColor(image.pixel(x, y))
            gray = int(0.299 * color.red() + 0.587 * color.green() + 0.114 * color.blue())
            sepia_r = min(255, int(gray * 1.2))
            sepia_g = min(255, int(gray * 1.0))
            sepia_b = min(255, int(gray * 0.8))
            image.setPixel(x, y, QColor(sepia_r, sepia_g, sepia_b).rgb())
    return QPixmap.fromImage(image)
```

パフォーマンスの考慮事項

メモリ使用量

- QPixmapは表示用に最適化されているため、メモリ使用量に注意
- 大きな画像は必要に応じてスケールダウン
- 不要になったQPixmapは適切に解放

描画パフォーマンス

- QPixmapはGPUで高速描画可能
- 頻繁に変更される画像にはQImageを使用検討
- キャッシュを活用して再描画を最小化

推奨事項

- アイコンには複数サイズを用意
- 高DPI対応として@2x画像も準備
- ファイル形式はPNG推奨（透明度対応）
- SVGアイコンも検討（スケーラブル）

実際のサンプルコード

`samples/q930_qpixmap_qicon/qpixmap_qicon_01.py` を参照してください。

関連クラス

- `QImage` - 画像データの直接操作用
- `QPainter` - 描画処理用
- `QBrush` - ブラシパターン用
- `QColor` - 色管理用