第 5 章

Graphics/View 绘图

在《编程改变生活——用 PySide6/PyQt6 创建 GUI 程序(基础篇 · 微课视频版)》的第 8 章介绍了使用 QPainter 类绘制图形的方法,这种方法比较适合绘制相对不复杂的图像,而且绘制的图形不能进行选择、编辑、拖放、修改。如果要绘制可交互的复杂图像,则应该怎么办?

为了绘制可交互的复杂图像,PySide6 提供了 Graphics/View 绘图框架。使用 Graphics/View 框架可绘制含有大量图形项(也称为图形元件)的图像,而且可以对每个图形项进行选择、拖放、修改等操作。

5.1 Graphics/View 简介

类比于将数据模型与视图控件相分离 Model/View 框架,Graphics/View 框架是将图像视图、图像场景、图形项相分离的框架,使用这样的技术可以绘制可交互的图像。具体来讲,主要使用了图像场景类 QGraphicsScene、图像视图类 QGraphicsView、图形项类 QGraphicsItem。

5.1.1 Graphics/View 绘图框架

在 PySide6 中,Graphics/View 绘图框架主要由图像视图、图像场景、图形项构成,这三者的系统结构如图 5-1 所示。

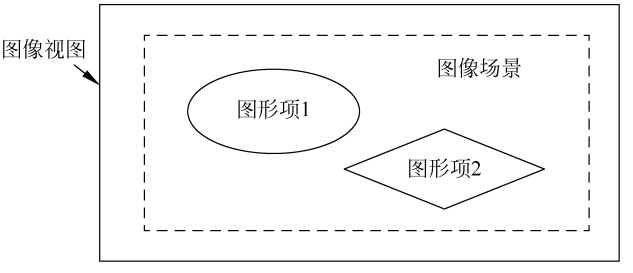

图 5-1 图像视图、图像场景、图形项的系统结构

1. 图像视图

图像视图类 QGraphicsView 提供了绘制图像的视图控件,用于显示图像场景中的内容。如果图像视图的范围大于图像场景的范围,则图像场景在图像视图中间部分显示;如果图像场景的范围大于图像视图的范围,则视图控件自动提供滚动条和滚动区。

QGraphicsView 类是视图控件,可以接受鼠标和键盘的输入并转换为场景事件,而且可以进行坐标转换后传递给可视的图像场景。

2. 图像场景

图像场景类 QGraphicsScene 提供了绘制图像的场景。图像场景是一个不可见的、抽象的容器,可以向图像场景中添加图形项,并可以获取图像场景中的各个图形项。

QGraphicsScene 类提供了大量的图形项接口,可以管理各个图形项及其状态,并可以将场景事件传递给各个图形项。

在实际编程中,可以设置图像场景背景色和前景色,主要使用了 QGraphicsScene 类的 drawBackground()和 drawForeground()方法。

3. 图形项

图形项就是一些基本的图形元件。图形项的基类为 QGraphicsItem,PySide6 也提供了标准的图形项类,例如矩形类 QGraphicsRectItem、椭圆类 QGraphicsEllipseItem、文本类 QGraphicsTextItem。

QGraphicsItem 类支持鼠标事件、键盘事件、拖放操作,也可以使用 QGraphicsItemGroup 类对图形元件进行组合,例如父子项关系组合。

综上所述,图像场景是图形项的容器,可以在图像场景中绘制多个图形项,每个图形项就是一个实例对象,这些图形项可以被选择、拖动。图像视图是显示图像场景的视图控件。一个图像场景可以有多张图像视图,一张图像视图可以显示图像场景的部分区域或全部区域。

5.1.2 Graphics/View 的坐标系

Graphics/View 框架有 3 个坐标系,分别是图像视图坐标系、图像场景坐标系、图形项坐标系。这 3 个坐标系的示意图如图 5-2 所示。

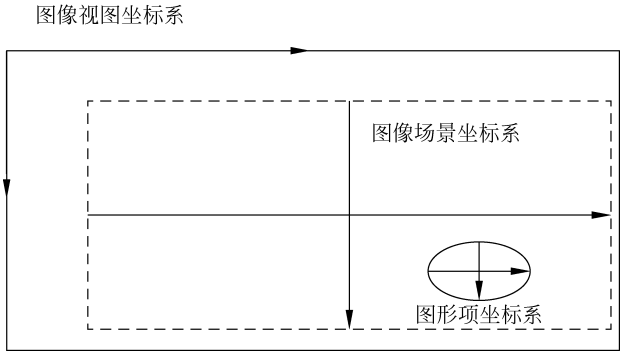

图 5-2 图像视图坐标系、图像场景坐标系、图形项坐标系的示意图

图像视图坐标系与设备坐标系相同，默认左上角为原点，这是物理坐标。图像场景坐标类系似于 QPainter 的逻辑坐标系，一般以图像场景的中心为坐标系原点（需要将图像场景的矩形范围设置为(－a，－b，2a，2b)，否则图像场景的中心点未必是坐标系原点），x 轴的正方向向右，y 轴正方向向下。图形项坐标系是局部的坐标系，通常以图形项的中心为坐标系原点，x 轴的正方向向右，y 轴的正方向向下。

1. 图像视图坐标

图像视图坐标是窗口控件的坐标，视图坐标的单位是像素。QGraphicsView 左上角的坐标为(0，0)。所有的鼠标事件、拖曳事件最开始都使用视图坐标。因为要和图形项交互，所以需要转换为图像场景坐标。

2. 图像场景坐标

图像场景坐标是所有图形项的基础坐标，场景坐标描述了顶层图形项的位置，而且构成了从图像视图到图像场景的所有场景事件的基础，每个图形项在场景上都有场景坐标和边界矩形。

3. 图形项坐标

图形项使用自己的局部坐标，通常以图形项的中心为原点。图形项的原点也是各种坐标转换的中心。图形项的鼠标事件使用局部坐标，创建图形项、绘制图形项也使用局部坐标，QGraphicsScene 和 QGraphicsView 会自动进行坐标转换。

一个图形项的位置是其中心点在父坐标系的坐标。如果一个图形项没有父图形项，则图形项的位置就是图像场景的坐标。如果一个图形项有父图形项，则父图形项进行坐标转换时子图形项也进行坐标转换。

4. 坐标变换

在 Graphics/View 框架下，经常需要在不同的坐标之间进行变换，例如从图像视图到图像场景、从图像场景到图形项、从子图形项到父图形项。Graphics/View 框架下的坐标变换方法见表 5-1。

表 5-1 Graphics/View 框架下的坐标变换方法

坐标变换方法	说 明
QGraphicsView. mapToScene()	从图像视图到图像场景
QGraphicsView. mapFromScene()	从图像场景到图像视图
QGraphicsItem. mapFromScene()	从图像场景到图形项
QGraphicsItem. mapToScene()	从图形项到图像场景
QGraphicsItem. mapToParent()	从子图形项到父图形项
QGraphicsItem. mapFromParent()	从父图形项到子图形项
QGraphicsItem. mapToItem()	从本图形项到其他图形项
QGraphicsItem. mapFromItem()	从其他图形项到本图形项

5.1.3 典型应用

下面使用 Graphics/View 框架绘制简单图像。

【实例 5-1】 使用 Graphics/View 框架绘制图像,该图像中包含一个矩形图形项、一个椭圆图形项,这两个图形项都可以移动,代码如下:

```
# === 第 5 章 代码 demo1.py === #
import sys
from PySide6.QtWidgets import (QApplication,QWidget,QGraphicsScene,
QGraphicsView,QVBoxLayout,QGraphicsRectItem,QGraphicsItem,QGraphicsEllipseItem)
from PySide6.QtCore import Qt,QRectF

class Window(QWidget):
    def __init__(self,parent = None):
        super().__init__(parent)
        self.setGeometry(200,200,580,280)
        self.setWindowTitle("使用 Graphics/View 绘图")
        #创建图像视图
        self.graphicsView = QGraphicsView()
        self.graphicsView.setBackgroundBrush(Qt.gray)
        #设置布局
        vbox = QVBoxLayout()
        vbox.addWidget(self.graphicsView)
        self.setLayout(vbox)
        #创建矩形范围
        rectF1 = QRectF( - 20, - 20,400,200)
        #创建图像场景
        self.graphicsScene = QGraphicsScene(rectF1)
        #图像视图设置图像场景
        self.graphicsView.setScene(self.graphicsScene)
        #以图像场景范围创建矩形图形项
        rectItem = QGraphicsRectItem(rectF1)
        rectItem.setBrush(Qt.yellow)
        rectItem.setFlags(QGraphicsItem.ItemIsSelectable|
QGraphicsItem.ItemIsMovable)
        #向图像场景中添加矩形
        self.graphicsScene.addItem(rectItem)
        #创建椭圆图形项
        rectF2 = QRectF( - 40, - 30,80,50)
        ellipseItem = QGraphicsEllipseItem(rectF2)
        ellipseItem.setBrush(Qt.red)        #设置画刷
        ellipseItem.setFlags(QGraphicsItem.ItemIsSelectable|
QGraphicsItem.ItemIsMovable)
        #向图像场景中添加椭圆图形项
        self.graphicsScene.addItem(ellipseItem)


if __name__ == "__main__":
    app = QApplication(sys.argv)
    win = Window()
    win.show()
    sys.exit(app.exec())
```

运行结果如图 5-3 所示。

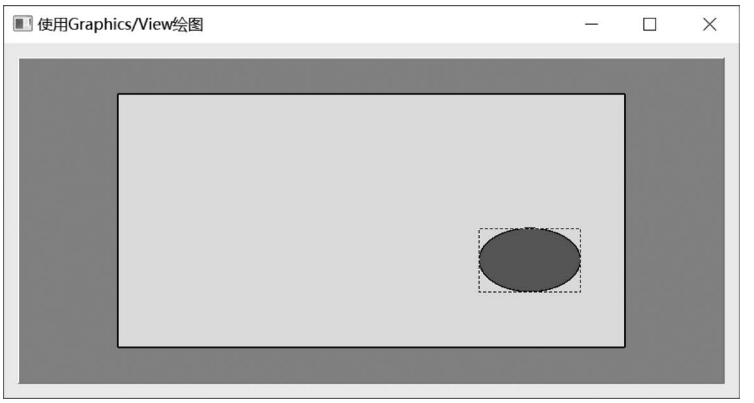

图 5-3　代码 demo1.py 的运行结果

【实例 5-2】 使用 Graphics/View 框架绘制图像，该图像中包含一个矩形图形项、一个椭圆图形项，这两个图形项都可以移动。当使用鼠标拖动图形项时，窗口状态栏显示鼠标的图像视图坐标、图像场景坐标、图形项坐标，需使用坐标变换的方法，代码如下：

```
# === 第 5 章 代码 demo2.py === #
import sys
from PySide6.QtWidgets import (QApplication,QMainWindow,QGraphicsScene,
QGraphicsView,QVBoxLayout,QStatusBar,QGraphicsRectItem,QGraphicsItem,QGraphicsEllipseItem)
from PySide6.QtCore import Qt,Signal,QPointF,QRectF

#创建图像视图控件的子类
class myGraphicsView(QGraphicsView):
    sendPosition = Signal(QPointF)      #自定义信号,参数是鼠标在视图中的位置
    def __init__(self,parent = None):
        super().__init__(parent)
    #鼠标单击事件
    def mousePressEvent(self,event):
        self.sendPosition.emit(event.scenePosition())      #发送信号,参数是鼠标位置
        super().mousePressEvent(event)
    #鼠标移动事件
    def mouseMoveEvent(self,event):
        self.sendPosition.emit(event.scenePosition())      #发送信号,参数是鼠标位置
        super().mouseMoveEvent(event)
    #重写背景函数,设置背景颜色
    def drawBackground(self, painter,rectF):
        painter.fillRect(rectF,Qt.gray)

class Window(QMainWindow):
    def __init__(self,parent = None):
        super().__init__(parent)
        self.setGeometry(200,200,580,280)
        self.setWindowTitle("坐标变换")
        #创建图像视图
        self.graphicsView = myGraphicsView()
        self.setCentralWidget(self.graphicsView)
```

```
        #创建状态栏
        self.statusbar = self.statusBar()
        rectF1 = QRectF(-200,-150,400,220)
        #创建图像场景
        self.graphicsScene = QGraphicsScene(rectF1)
        #图像视图设置图像场景
        self.graphicsView.setScene(self.graphicsScene)
        #创建矩形图形项
        rectItem = QGraphicsRectItem(rectF1)
        rectItem.setFlags(QGraphicsItem.ItemIsSelectable
|QGraphicsItem.ItemIsMovable)                              #设置标识
        #向图像场景中添加矩形图形项
        self.graphicsScene.addItem(rectItem)
        rectF2 = QRectF(-40,-40,120,80)
        #创建椭圆图形项
        ellipseItem = QGraphicsEllipseItem(rectF2)
        #设置画刷
        ellipseItem.setBrush(Qt.red)
        ellipseItem.setFlags(QGraphicsItem.ItemIsSelectable
|QGraphicsItem.ItemIsMovable)
        #向图像场景中添加椭圆图形项
        self.graphicsScene.addItem(ellipseItem)
        #使用信号/槽
        self.graphicsView.sendPosition.connect(self.mousePosition)
    #自定义槽函数
    def mousePosition(self,pointF):
        point = pointF.toPoint()
        template = "视图坐标:{},{} 场景坐标:{},{} 图形项坐标:{},{}"
        #将视图中的点映射到场景中
        pointScene = self.graphicsView.mapToScene(point)
        #第 1 种获取视图控件中的图形项的方法
        item = self.graphicsView.itemAt(point)
        #第 2 种获取视图控件中的图形项的方法
        #item = self.graphicsScene.itemAt(pointScene,
self.graphicsView.transform())
        if item:
            pointItem = item.mapFromScene(pointScene)      #把场景坐标转换为图形项坐标
            string = template.format(point.x(),point.y(),pointScene.x(),
pointScene.y(),pointItem.x(),pointItem.y())
        else:
            string = template.format(point.x(), point.y(),pointScene.x(),
pointScene.y(),"None","None")
        self.statusbar.showMessage(string)                 #在状态栏中显示坐标信息


if __name__ == "__main__":
    app = QApplication(sys.argv)
    win = Window()
    win.show()
    sys.exit(app.exec())
```

运行结果如图 5-4 所示。

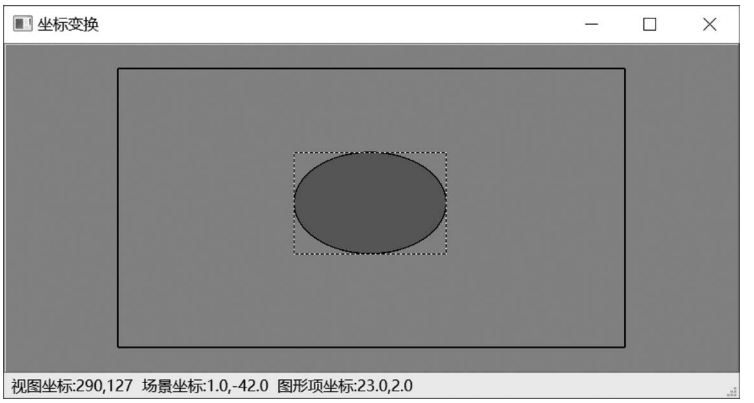

图 5-4　代码 demo2.py 的运行结果

5.2　Graphics/View 相关类

使用 Graphics/View 框架绘制可交互图像主要使用了图像视图类 QGraphicsView、图像场景类 QGraphicsScene、图形项类 QGraphicsItem、标准图形项类。本节主要介绍这 4 种类的构造函数和常用方法。

5.2.1　图像视图类 QGraphicsView

图像视图类 QGraphicsView 是 QAbstractScrollAren 类的子类。QGraphicsView 类创建的图像视图控件可以根据图像场景的宽和高提供滚动区，当图像视图控件的宽和高小于图像场景的宽和高时会提供滚动条。QGraphicsView 类的继承关系如图 5-5 所示。

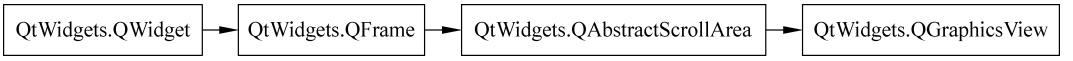

图 5-5　QGraphicsView 类的继承关系

QGraphicsView 类位于 PySide6 的 QtWidgets 子模块下，其构造函数如下：

```
QGraphicsView(parent:QWidget = None)
QGraphicsView(scene:QGraphicsScene,parent:QWidget = None)
```

其中，parent 表示父窗口或父容器；scene 表示图像场景对象。

QGraphicsView 类的常用方法见表 5-2。

表 5-2　QGraphicsView 类的常用方法

方法及参数类型	说　明	返回值的类型
[slot]updateScene(rects:Sequence[QRectF])	更新场景	None
[slot]updateSceneRect(rect:Union[QRectF,QRect])	更新场景	None

续表

方法及参数类型	说　明	返回值的类型
[slot] invalidateScene(rect: Union[QRectF, QRect], layers: QGraphicsScene.SceneLayers = QGraphicsScene.AllLayers)	对指定的场景区域进行更新和重绘，相当于对指定区域进行 update() 操作	None
setScene(scene:QGraphicsScene)	设置图像场景	None
scene()	获取图像场景	QGraphicsScene
setSceneRect(rect:Union[QRectF,QRect])	设置图像场景在图像视图中的范围	None
setSceneRect(x:float,y:float,w:float,h:float)	设置图像场景在图像视图中的范围	None
sceneRect()	获取图像场景在图像视图中的范围	QRectF
setAlignment(Qt.Alignment)	设置图像场景全部可见时的对齐方式	None
setBackgroundBrush(brush: Union[QBrush, Qt.BrushStyle, Qt.GlobalColor, QColor, QGradient, QImage, QPixmap])	设置背景色	None
setForegroundBrush(brush: Union[QBrush, Qt.BrushStyle, Qt.GlobalColor, QColor, QGradient, QImage, QPixmap])	设置前景色	None
drawBackground(painter:QPainter,rect:Union[QRectF,QRect])	重写该函数，在显示前景和图形项前绘制背景	None
drawForeground(painter:QPainter,rect:Union[QRectF,QRect])	重写该函数，在显示前景和图形项后绘制背景	None
centerOn(pos:Union[QPointF,QPoint,QPainterPath.Element])	使某个点位于视图控件中心	None
centerOn(x:float,y:float)	使某个点位于视图控件中心	None
centerOn(item:QGraphicsItem)	使某个图形项位于视图控件中心	None
ensureVisible(rect: Union[QRectF, QRect], xmargin:int=50,ymargin:int=50)	确保在指定的矩形区域可见，若可见，则按照指定的边距显示；若不可见，则滚动到最近的点	None
ensureVisible(x: float, y: float, w: float, h: float,xmargin:int=50,ymargin:int=50)		None
ensureVisible(QGraphicsItem, xmargin: int = 50,ymargin:int=50)	确保指定的图形项可见	None
fitInView(rect: Union[QRectF, QRect], aspectRadioMode:Qt.AspectRadioMode=Qt.IngnoreAspectRadio)	以合适的方式使矩形区域可见	None
fitInView(x: float, y: float, w: float, h: float, aspectRadioMode:Qt.AspectRadioMode=Qt.IngnoreAspectRadio)	以合适的方式使矩形区域可见	None

续表

方法及参数类型	说　明	返回值的类型
fitInView(item: QGraphicsItem, aspectRadioMode: Qt. AspectRadioMode=Qt. IngnoreAspectRadio)	以合适的方式使图形项可见	None
render(painter: QPainter, target: Union[QRectF, QRect], source: QRect, aspectRatioMode=Qt. KeepAspectRadio)	将图像从 source(视图控件)复制到 target(其他设备,如 QImage)上	None
resetCachedContent()	重置缓存	None
rubberBandRect()	获取用鼠标框选的范围	QRect
setCacheMode(mode: QGraphicsView. CacheMode)	设置缓存模式	None
setDragMode(mode: QGraphicsView. DragMode)	设置鼠标拖曳模式	None
setInteractive(allowed: bool)	设置是否为交互模式	None
isInteractive()	获取是否为交互模式	bool
setOptimizationFlag(flag: QGraphicsView. OptimizationFlag, enabled: bool=True)	设置优化显示标识	None
setOptimizationFlags(flags: QGraphicsView. OptimizationFlags)	设置优化显示标识	None
setRenderHint(hint: QPainter. RenderHint, enabled: bool=True)	设置提供绘图质量的标识	None
setRenderHints(hint: QPainter. RenderHints)	设置提供绘图质量的标识	None
setResizeAnchor(QGraphicsView. ViewportAnchor)	设置视图控件改变宽和高时的锚点	None
resizeAnchor()	获取锚点	ViewportAnchor
setRubberBandSelectionMode(Qt. ItemSelectionMode)	设置鼠标框选模式	None
setTransform(matrix: QTransform, combine: bool=False)	用变换矩阵变换视图	None
transform()	获取变换矩阵	QTransform
isTransformed()	获取是否进行过变换	bool
resetTransform()	重置变换	None
setTransformationAnchor(QGraphicsView. ViewportAnchor)	设置变换时的锚点	None
setViewportUpdateMode(QGraphicsView. ViewportUpdateMode)	设置刷新模式	None
setupViewport(QWidget)	重写该函数,设置视口控件	None
scale(sx: float, sy: float)	缩放	None
shear(sh: float, sv: float)	错切	None
rotate(angle: float)	旋转,顺时针方向为正	None
translate(dx: float, dy: float)	平移	None

在表5-2中,Qt.Alignment的枚举值为Qt.AlignLeft、Qt.AlignRight、Qt.AlignHCenter、Qt.AlignJustify、Qt.AlignTop、Qt.AlignBottom、Qt.AlignVCenter、Qt.AlignBaseline、Qt.AlignCenter(默认值)。

QGraphicsView.CacheMode的枚举值为QGraphicsView.CacheNone(没有缓存)、QGraphicsView.CacheBackground(缓存背景)。

QGraphicsView.DragMode的枚举值为QGraphicsView.NoDrag(忽略鼠标事件)、QGraphicsView.ScrollHandDrag(在交互或非交互模式下,光标变成手形状,拖动鼠标会移动图像场景)、QGraphicsView.RubberBandDrag(在交互模式下,可以框选图形项)。

Qt.ItemSelectionMode的枚举值为Qt.ContainsItemShape、Qt.IntersectsItemShape、Qt.ContainsItemBoundingRect、Qt.IntersectsItemBoundingRect。

QGraphicsView.OptimizationFlag的枚举值为QGraphicsView.DontSavePainterState(不保存绘图状态)、QGraphicsView.DontAdjustForAntialiasing(不调整反锯齿)、QGraphicsView.IndirectPainting(间接绘制)。

QGraphicsView.ViewportAnchor的枚举值为QGraphicsView.NoAnhor(没有锚点,场景位置不变)、QGraphicsView.AnchorViewCenter(场景将视图控件的中心点作为锚点)、QGraphicsView.AnchorUnderMouse(将光标所在的位置作为锚点)。

QGraphicsView.ViewportUpdateMode的枚举值为QGraphicsView.FullViewportUpdate、QGraphicsView.MinimalViewportUpdate、QGraphicsView.SmartViewportUpdate、QGraphicsView.BoundingRectViewPortUpdate、QGraphicsView.NoViewportUpdate。

QGraphicsScene.SceneLayers的枚举值为QGraphicsScene.ItemLayer、QGraphicsScene.BackgroundLayer、QGraphicsScene.ForegroundLayer、QGraphicsScene.AllLayers。

QGraphicsView类获取图形项的方法见表5-3。

表5-3 QGraphicsView类获取图形项的方法

方法及参数类型	返回值类型
itemAt(pos:QPoint)	QGraphicsItem
itemAt(x:int,y:int)	QGraphicsItem
items()	List[QGraphicsItem]
items(pos:QPoint)	List[QGraphicsItem]
items(x:int,y:int)	List[QGraphicsItem]
items(x:int,y:int,w:int,h:int,mode=Qt.intersectsItemShape)	List[QGraphicsItem]
items(rect:QRect,mode:Qt.ItemSelectionMode=Qt.IntersectsItemShape)	List[QGraphicsItem]
items(polygon: Union[QPolygon, Sequence[QPoint], QRect], mode: Qt.ItemSelectionMode=Qt.IntersectsItemShape)	List[QGraphicsItem]
items(QPainterPath,mode:Qt.ItemSelectionMode=Qt.IntersectsItemShape)	List[QGraphicsItem]

Qt.IntersectsItemShape的枚举值为Qt.ContainsItemShape(图形项完全在选择框的内部)、Qt.IntersectsItemShape(图形项在选择框的内部与选择框相交)、Qt.ContainsItem-

BoundingRect(图形项的边界矩形完全在选择框的内部)、Qt. IntersectsItemBoundingRect(图形项的边界矩形在选择框的内部与选择框交叉)。

QGraphicsView 类中将图像视图坐标转换为图像场景坐标的方法见表 5-4。

表 5-4 QGraphicsView 类中将图像视图坐标转换为图像场景坐标的方法

方法及参数类型	返回值类型
mapToScene(point:QPoint)	QPointF
mapToScene(rect:QRect)	QPolygonF
mapToScene(Union[QPolygon,Sequence[QPoint],QRect])	QPolygonF
mapToScene(QPainterPath)	QPainterPath
mapToScene(x:int,y:int)	QPointF
mapToScene(int,int,int,int)	QPolygonF

QGraphicsView 类中将图像场景坐标转换为图像视图坐标的方法见表 5-5。

表 5-5 QGraphicsView 类中将图像场景坐标转换为图像视图坐标的方法

方法及参数类型	返回值类型
mapFromScene(Union[QPointF,QPoint])	QPoint
mapFromScene(QRectF)	QPolygon
mapFromScene(polygon: Union[QPolygonF, Sequence[QPointF], QPolygon, QRectF])	QPolygon
mapFromScene(path:QPainterPath)	QPainterPath
mapFromScene(x:float,y:float)	QPoint
mapFromScene(x:float,y:float,w:float,h:float)	QPolygon

QGraphicsView 类只有一个信号 rubberBandChanged(viewportRect: QRect, fromScenePoint:QPointF,toScenePoint:QPointF),表示当框选范围发生改变时发送信号。

5.2.2 图像场景类 QGraphicsScene

在 PySide6 中,使用 QGraphicsScene 类创建图像场景对象。图像场景对象是存放图形项的容器,用于存放和管理图形项。QGraphicsScene 类是 QObject 类的子类,位于 QtWidgets 子模块下,其构造函数如下:

```
QGraphicsScene(parent:QObject = None)
QGraphicsScene(sceneRect:Union[QRectF,QRect],parent:QObject = None)
QGraphicsScene(x:float,y:float,width:float,height:float,parent:QObject = None)
```

其中,parent 表示 QObject 类及其子类创建的实例对象; sceneRect 表示场景的范围。如果未设置场景的范围,则可以使用 sceneRect()方法获取图像场景中包含图形项的最大矩形边界。当在图像场景中添加、移动图形项时,场景范围会增大,但不会减小。

在 QGraphicsScene 类中添加和移除图形项的方法见表 5-6。

表 5-6 QGraphicsScene 类中添加和移除图形项的方法

方法及参数类型	说 明	返回值的类型
[slot]clear()	清空所有图形项	None
addItem(QGraphicsItem)	添加图形项	None
addEllipse(rect: Union[QRectF, QRect], pen: Union[QPen, Qt. PenStyle, QColor], brush: Union[QBrush, Qt. BrushStyle, Qt. GlobalColor, QColor, QGradient, QImage, QPixmap])	添加椭圆	QGraphicsEllipseItem
addEllipse(x:float, y:float, w:float, h:float, pen, brush)	添加椭圆	QGraphicsEllipseItem
addLine(line: Union[QLineF, QLine], pen)	添加线段	QGraphicsLineItem
addLine(x1:float, y1:float, x2:float, y2:float, pen)	添加线段	QGraphicsLineItem
addPath(path: QPainterPath, pen, brush)	添加绘图路径	QGraphicsPathItem
addPixmap(pixmap: Union[QPixmap, QImage, str])	添加图像	QGraphicsPixmapItem
addPolygon(polygon: Union[QPolygonF, Sequence[QPointF], QPolygon, QRectF], pen, brush)	添加多边形	QGraphicsPolygonItem
addRect(rect: Union[QRectF, QRect], pen, brush)	添加矩形	QGraphicsRectItem
addRect(x:float, y:float, w:float, h:float, pen, brush)	添加矩形	QGraphicsRectItem
addSimpleText(text: str, font: Union[QFont, str])	添加简单文字	QGraphicsSimpleTextItem
addText(text: str, font: Union[QFont, str])	添加文字	QGraphicTextItem
addWidget(QWidget, wFlags: Qt. WindowFlags)	添加控件	QGraphicsProxyWidget
removeItem(QGraphicsItem)	移除指定图形项	None

QGraphicsScene 类中获取图形项的方法见表 5-7。

表 5-7 QGraphicsScene 类中获取图形项的方法

方法及参数类型	返回值类型
itemAt(pos: Union[QPointF, QPoint, QPainterPath. Element], deviceTransform: QTransform)	QGraphicsItem
itemAt(x:float, y:float, deviceTransform: QTransform)	QGraphicsItem
items(order: Qt. SortOrder = Qt. DescendingOrder)	List[QGraphicsItem]
items(path: QPainterPath, mode: Qt. ItemSelectionMode = Qt. IntersectsItemShape, order, deviceTransform)	List[QGraphicsItem]
items(polygon: Union[QPolygonF, Sequence[QPointF], QPolygon, QRectF], mode, order, deviceTransform)	List[QGraphicsItem]
items(pos: Union[QPointF, QPoint, QPainterPath. Element], mode, order, deviceTransform)	List[QGraphicsItem]
items(rect: Union[QRectF, QRect], mode, order, deviceTransform)	List[QGraphicsItem]
items(x:float, y:float, w:float, h:float, mode, order, deviceTransform)	List[QGraphicsItem]

其中,mode 的参数值为 Qt. ItemSelectionMode 的枚举值。Qt. ItemSelectionMode 的枚举值为 Qt. ContainsItemShape(完全包含)、Qt. IntersectsItemShape(包含和交叉)、Qt. ContainsItemBoundingRect(完全包含矩形边界)、Qt. IntersectsItemBoundingRect(包

含矩形边界和交叉边界)。order 的参数值为 Qt. DescendingOrder(降序)、Qt. AscendingOrder(升序)。

QGraphicsScene 类的其他常用方法见表 5-8。

表 5-8 QGraphicsScene 类的其他常用方法

方法及参数类型	说 明	返回值的类型
[slot]advance()	调用图形项的 advance()方法,通知图形项可移动	None
[slot]clearSelection()	取消选择	None
[slot] invalidate (rect: Union [QRectF, QRect], Layers=QGraphicsScene. AllLayers)	刷新指定的区域	None
invalidate(x:float, y:float, w:float, h:float, Layers=QGraphicsScene. AllLayers)	刷新指定的区域	None
[slot]update(rect:Union[QRectF,QRect])	更新区域	None
update(x:float,y:float,w:float,h:float)	更新区域	None
setSceneRect(rect:Union[QRectF,QRect])	设置场景范围	None
setSceneRect(x: float, y: float, w: float, h: float)	设置场景范围	None
sceneRect()	获取场景范围	QRectF
width()	获取场景的宽度	float
height()	获取场景的高度	float
collidingItems (QGraphicsItem, mode: Qt. IntersectsItemShape)	获取碰撞的图形项列表	List[QGraphicsItem]
createItemGroup(Sequence[QGraphicsItem])	创建图形项组合	QGraphicsItemGroup
destroyItemGroup(QGraphicsItemGroup)	打散图形项组合	None
hasFocus()	获取图像场景是否有焦点,若有焦点,则可接受键盘事件	bool
clearFocus()	清除场景中的焦点	None
isActive()	若图像场景在视图控件中显示并且视图控件活跃时,则返回值为 True	bool
itemsBoundingRect()	获取图形项的矩形区域	QRectF
mouseGrabberItem()	获取光标抓取的图形项	QGraphicItem
render (QPainter, target: QRectF, source: QRectF,mode:Qt. KeepAspectRadio)	将指定区域的图形复制到其他设备的指定区域上	None
selectedItems()	获取选中的图形项列表	List[QGraphicsItem]
setActivePanel(item:QGraphicsItem)	将场景中的图形项设置为活跃图形项	None
activePanel()	获取活跃的图形项	None
setActiveWindow(widget:QGraphicsWidget)	将场景中的视图控件设置为活跃控件	None

续表

方法及参数类型	说　明	返回值的类型
setBackgroundBrush(Union[QBrush, QColor, Qt.GlobalColor, QGradient])	设置背景画刷	None
setForegroundBrush(Union[QBrush, QColor, Qt.GlobalColor, QGradient])	设置前景画刷	None
drawBackground(QPainter, QRectF)	重写该函数,绘制背景	None
drawForeground(QPainter, QRectF)	重写该函数,绘制前景	None
backgroundBrush()	获取背景画刷	QBrush
foregroundBrush()	获取前景画刷	QBrush
setFocus(focusReason=Qt.OtherFocusReason)	设置图像场景获得焦点	None
setFocusItem(QGraphicsItem, focusReason: Qt.FocusReason=Qt.OtherFocusReason)	设置某个图形项获得焦点	None
focusItem()	获取有焦点的图形项	QGraphicItem
setFocusOnTouch(bool)	在平板电脑上设置是否通过手触碰获得焦点	None
focusNextPrevChild(next:bool)	查找一个新的图形控件,并使键盘焦点(例如 Tab 键、Shift+Tab 键)对准该图形项,若找到,则返回值为 True。若 next 的值为 True,则向前搜索,否则向后搜索	bool
setItemIndexMethod(QGraphicsScene.ItemIndexMethod)	设置图形项搜索方法	None
setBspTreeDepth(int)	设置 BSP 树的搜索深度	None
setMinimumRenderSize(float)	图形项变换后,若图形项的宽和高小于设置的宽和高,则不渲染	None
setSelectionArea(path:QPainterPath, deviceTransform)	选择绘图路径内的图形项,绘图路径外的图形项取消选中。对于需要选中的图形项,必须标记为 QGraphicsItem.ItemIsSelectable	None
setSelectionArea(path: QPainterPath, selectionOperation: Qt.ItemSelectionOperation=Qt.ReplaceSelection, mode: Qt.ItemSelectionMode = Qt.IntersectsItemShapedeviceTransform:QtTransform=Default(QTransform))		None
selectionArea()	获取选择区域内的绘图路径	QPainterPath
setStickyFocus(enabled:bool)	当单击背景或不接受焦点的图形项时,设置是否失去焦点	None
setFont(QFont)	设置字体	None
setPalette(QPalette)	设置调色板	None
setStyle(QStyle)	设置风格	None
views()	获取与场景关联的视图控件列表	List[QGraphicsItem]

在表 5-8 中，QGraphicsScene. ItemIndexMethod 的枚举值为 QGraphicsScene. BspTreeIndex（BSP 树方法，适合静态场景）、QGraphicsScene. NoIndex（适合动态场景）。

QGraphicsScene 类的信号见表 5-9。

表 5-9 QGraphicsScene 类的信号

信号及参数类型	说 明
changed(region:List[QRectF])	当图像场景中的内容发生改变时发送信号，参数为包含场景的矩形列表，这些矩形表示已更改的区域
focusItemChanged(newFocusItem:QGraphicsItem, oldFocusItem:QGraphicsItem, reason:Qt. FocusReason)	当图形项的焦点改变时，或者焦点从一个图形项转移到另一个图形项时，发送信号
sceneRectChanged(rect:QRectF)	当图像场景的范围发生改变时发送信号
selectionChanged()	当图像场景中被选中的图形项发生改变时发送信号

5.2.3 图形项类 QGraphicsItem

在 PySide6 中，QGraphicsItem 类是所有图形项类的基类。可以使用 QGraphicsItem 类创建自定义图形项类，包括定义几何形状、碰撞检测、绘图实现，以及通过事件处理函数进行图形项的交互。图形项支持鼠标事件、滚轮事件、键盘事件，如果进行分组和碰撞检测，则可以给图形项设置数据。

QGraphicsItem 类位于 PySide6 的 QtWidgets 子模块下，其构造函数如下：

```
QGraphicsItem(parent:QGraphicsItem = None)
```

其中，parent 表示父图形项，数据类型为 QGraphicsItem 对象。

QGraphicsItem 类的常用方法见表 5-10。

表 5-10 QGraphicsItem 类的常用方法

方法及参数类型	说 明	返回值的类型
childItem()	获取子项列表	List[QGraphicsItem]
childrenBoundingRect()	获取子项的边界矩形	QRectF
clearFocus()	清除焦点	None
paint(painter:QPainter, option:QStyleOptionGraphicsItem, widget:QWidget=None)	重写该函数，绘制图形	None
boundingRect()	重写该函数，获取边界矩形	QRectF
itemChange(change:QGraphicsItem. GraphicsItemChange, value:Any)	重写该函数，以使当图形项状态发生改变时作出响应	None

续表

方法及参数类型	说　明	返回值的类型
advance(phase)	重写该函数,用于简单动画,由场景的advance()调用。若phase=0,则通知图形项即将运动;若phase=1,则可以运动	None
setCacheMode(mode:QGraphicsItem.CacheMode,casheSize:QSize=Default(QSize))	设置图形项的缓冲模式	None
collidesWithItem(other:QGraphicsItem,mode:Qt.ItemSelectionMode=Qt.IntersectsItemShape)	获取是否能与指定的图形项发生碰撞	bool
collidesWithPath(path:QPainterPath,mode:Qt.ItemSelectionMode=Qt.IntersectsItemShape)	获取是否能与指定的路径发生碰撞	bool
collidingItems(mode=Qt.IntersectsItemShape)	获取能发生碰撞的图形项列表	List[QGraphicsItem]
contains(Union[QPointF,QPoint])	获取图形项是否包含某个点	bool
grabKeyboard()	接受键盘的所有事件	None
unGrabKeyboard()	不接受键盘的所有事件	None
grabMouse()	接受鼠标的所有事件	None
unGrabMouse()	不接受鼠标的所有事件	None
isActive()	获取图形项是否活跃	bool
isAncestorOf(QGraphicsItem)	获取图形项是否为指定图形项的父辈	bool
isEnabled()	获取是否激活	bool
isPanel()	获取是否为面板	bool
isSelected()	获取是否被选中	bool
isUnderMouse()	获取是否位于光标下	bool
parentItem()	获取父图形项	QGraphicsItem
resetTransform()	重置变换	None
scene()	获取图形项所在的场景	QGraphicsScene
sceneBoundingRect()	获取场景的范围	QRectF
scenePos()	获取在场景中的位置	QPointF
sceneTransform()	获取变换矩阵	QTransform
setAcceptDrops(bool)	设置鼠标是否接受鼠标释放事件	None
setAcceptedMouseButtons(Qt.MouseButton)	设置可接受的鼠标按钮	None
setActive(bool)	设置是否活跃	None

续表

方法及参数类型	说　明	返回值的类型
setCursor(Union[QCursor, Qt.CursorShape])	设置光标形状	None
unsetCursor()	重置光标形状	None
setData(key:int,value:int)	设置图形项的数据	None
data(key:int)	获取图形项存储的数据	object
setEnabled(bool)	设置图形项是否激活	None
setFlag(QGraphicsItem.GraphicsItemFlag,enable=True)	设置图形项的标识	None
setFocus(focusReason=Qt.OtherFocusReason)	设置焦点	None
setGroup(QGraphicsItemGroup)	将图形项加入组合中	None
group()	获取图形项所在的组合	QGraphicsItemGroup
setOpacity(opacity:float)	设置不透明度	None
setPanelModality(QGraphicsItem.PanelModality)	设置面板的模式	None
setParentItem(QGraphicsItem)	设置父图形项	None
setPos(Union[QPointF,QPoint])	设置在父图形项坐标系中的位置	None
setPos(x:float,y:float)	设置在父图形项坐标系中的位置	None
setX(float)	设置在图形项中的 x 坐标	None
setY(float)	设置在图形项中的 y 坐标	None
pos()	获取图形项在父图形项中的位置	QPointF
x()、y()	获取 x 坐标、获取 y 坐标	float
setRotation(angle:float)	设置沿 z 轴顺时针旋转角度(角度值)	None
setScale(scale:float)	设置缩放比例系数	None
moveBy(dx:float,dy:float)	设置移动量	None
setSelected(selected:bool)	设置是否被选中	None
setToolTip(str)	设置提示信息	None
setTransform(QTransform,combine=False)	设置矩阵变换	None
setTransformOriginPoint(origin:Union[QPointF,QPoint])	设置变换的中心点	None
setTransformOriginPoint(ax:float,ay:float)	设置变换的中心点	None
setTransformations(Sequence[QGraphicsTransform])	设置变换矩阵	None
transform()	获取变换矩阵	QTransform
transformOriginPoint()	获取变换原点	QPointF
setVisible(bool)	设置图形项是否可见	None
show()	显示图形项	None

续表

方法及参数类型	说　明	返回值的类型
hide()	隐藏图形项,包括子图形项	None
isVisible()	获取是否可见	None
setZValue(float)	设置 z 值	None
zValue()	获取 z 值	float
shape()	重写该函数,获取图形项的绘图路径,用于碰撞检测	QPainterPath
stackBefore(QGraphicsItem)	在指定的图形项之前插入	None
isWidget()	获取图形项是否为图形控件 QGraphicsWidget	bool
isWindow()	获取图形控件的窗口类型是否为 Qt. Window	bool
window()	获取图形项所在的图形控件	QGraphicsWidget
topLevelWidget()	获取顶层图形控件	QGraphicsWidget
topLevelItem()	获取顶层图形项,即没有父图形项的图形项	QGraphicsItem
update(rect:Union[QRectF, QRect]=Default(QRectF))	更新指定区域	None
update(x: float, y: float, width: float,height:float)	更新指定区域	None

QGraphicsItem 类的坐标映射方法见表 5-11。

表 5-11　QGraphicsItem 类的坐标映射方法

类　型	方法及参数类型	返回值类型
从其他图形项映射	mapFromItem(item:QGraphicsItem,path:QPainterPath)	QPainterPath
	mapFromItem(item: QGraphicsItem, point: Union[QPointF, QPoint])	QPointF
	mapFromItem(item: QGraphicsItem, polygon: Union[QPolygonF, Sequence[QPointF],QPolygon,QRectF])	QPolygonF
	mapFromItem(item: QGraphicsItem, rect: Union[QRectF, QRect])	QPolygonF
	mapFromItem(item:QGraphicsItem,x:float,y:float)	QPointF
	mapFromItem(item:QGraphicsItem,x:float,y:float,w:float,h:float)	QPolygonF
	mapFromRectItem(item:QGraphicsItem,rect:Union[QRectF,QRect])	QRectF
	mapFromRectItem(item: QGraphicsItem, x: float, y: float, w: float,h:float)	QRectF

续表

类　型	方法及参数类型	返回值类型
从父图形项映射	mapFromParent(path:QPainterPath)	QPainterPath
	mapFromParent(point:Union[QPoint,QPointF])	QPointF
	mapFromParent(polygon:Union[QPolygonF,Sequence[QPointF],QPolygon,QRectF])	QPolygonF
	mapFromParent(rect:Union[QRectF,QRect])	QPolygonF
	mapFromParent(x:float,y:float,w:float,h:float)	QPolygonF
	mapFromParent(x:float,y:float)	QPointF
	mapRectFromParent(rect:Union[QRectF,QRect])	QRectF
	mapRectFromParent(x:float,y:float,w:float,h:float)	QRectF
从图像场景映射	mapFromScene(path:QPainterPath)	QPainterPath
	mapFromScene(point:Union[QPoint,QPointF])	QPointF
	mapFromScene(polygon:Union[QPolygonF,Sequence[QPointF],QPolygon,QRectF])	QPolygonF
	mapFromScene(rect:Union[QRectF,QRect])	QPolygonF
	mapFromScene(x:float,y:float,w:float,h:float)	QPolygonF
	mapFromScene(x:float,y:float)	QPointF
	mapRectFromScene(rect:Union[QRectF,QRect])	QRectF
	mapRectFromScene(x:float,y:float,w:float,h:float)	QRectF
映射到其他图形项	mapToItem(item:QGraphicsItem,path:QPainterPath)	QPainterPath
	mapToItem(item:QGraphicsItem,point:Union[QPointF,QPoint])	QPointF
	mapToItem(item:QGraphicsItem,polygon:Union[QPolygonF,Sequence[QPointF],QPolygon,QRectF])	QPolygonF
	mapToItem(item:QGraphicsItem,rect:Union[QRectF,QRect])	QPolygonF
	mapToItem(item:QGraphicsItem,x:float,y:float)	QPointF
	mapToItem(item:QGraphicsItem,x:float,y:float,w:float,h:float)	QPolygonF
	mapRectToItem(item:QGraphicsItem,rect:Union[QRectF,QRect])	QRectF
	mapRectToItem(item:QGraphicsItem,x:float,y:float,w:float,h:float)	QRectF
映射到父图形项	mapToParent(path:QPainterPath)	QPainterPath
	mapToParent(point:Union[QPoint,QPointF])	QPointF
	mapToParent(polygon:Union[QPolygonF,Sequence[QPointF],QPolygon,QRectF])	QPolygonF
	mapToParent(rect:Union[QRectF,QRect])	QPolygonF
	mapToParent(x:float,y:float,w:float,h:float)	QPolygonF
	mapToParent(x:float,y:float)	QPointF
	mapRectToParent(rect:Union[QRectF,QRect])	QRectF
	mapRectToParent(x:float,y:float,w:float,h:float)	QRectF

续表

类　型	方法及参数类型	返回值类型
映射到图像场景	mapToScene(path:QPainterPath)	QPainterPath
	mapToScene(point:Union[QPoint,QPointF])	QPointF
	mapToScene(polygon:Union[QPolygonF,Sequence[QPointF],QPolygon,QRectF])	QPolygonF
	mapToScene(rect:Union[QRectF,QRect])	QPolygonF
	mapToScene(x:float,y:float,w:float,h:float)	QPolygonF
	mapToScene(x:float,y:float)	QPointF
	mapRectToScene(rect:Union[QRectF,QRect])	QRectF
	mapRectToScene(x:float,y:float,w:float,h:float)	QRectF

在 QGraphicsItem 类中,使用 setFlag(QGraphicsItem. GraphicsItemFlag,enabled=True)方法设置图形项的标志,其中参数 QGraphicsItem. GraphicsItemFlag 的枚举值见表 5-12。

表 5-12　QGraphicsItem. GraphicsItemFlag 的枚举值

枚　举　值	说　明
QGraphicsItem. ItemIsMovable	可移动
QGraphicsItem. ItemSelectable	可选择
QGraphicsItem. ItemIsFocusable	可获得键盘输入焦点、鼠标按下、鼠标释放事件
QGraphicsItem. ItemClipsToShape	剪切自己的图形项,在图形项之外不能接受鼠标拖放和悬停事件
QGraphicsItem. ItemClipsChildrenToShape	剪切子类的图形项,子类不能在该图形项之外绘制
QGraphicsItem. ItemIgnoresTransformations	忽略来自父图形项和视图控件的坐标变换,例如文字保持水平或竖直,文字比例不缩放
QGraphicsItem. ItemIgnoreParentOpacity	使用自身的透明设置,不使用父图形项的透明设置
QGraphicsItem. ItemDoesntPropagateOpacityToChildren	图形项的透明设置不影响子图形项的透明值
QGraphicsItem. ItemStacksBehindParent	放置在父图形项的后面,而不是前面
QGraphicsItem. ItemHasNoContents	在图形项中不绘制任何图形,调用 paint()方法也不起作用
QGraphicsItem. ItemSendsGeometryChanges	该标志可用 itemsChange()方法处理图形项几何形状的改变,例如 ItemPositionChange、ItemScaleChange、ItemPositionHasChanged、ItemTransformChange、ItemTransformHasChanged、ItemRotationChange、ItemRotationHasChanged、ItemScaleHasChanged、ItemTransformOriginPointChange、ItemTransformOriginPointHasChange
QGraphicsItem. ItemAcceptInputMethod	图形项支持亚洲语言
QGraphicsItem. ItemNegativeZStacksBehindParent	若图形项的 z 值为负值,则自动放置在父图形项的后面,可使用 setZValue()方法切换图形项与父图形项的位置

续表

枚 举 值	说 明
QGraphicsItem. ItemIsPanel	图形项为面板，面板可被激活，获得焦点。在同一时间只有一个面板能被激活，若没有面板，则激活所有非面板图形项
QGraphicsItem. ItemSendsScenePositonChange	该标志可用 itemChange()方法处理图形项在视图控件中的位置变化事件 ItemScenePositonHasChanged
QGraphicsItem. ItemContainsChildrenInShape	该标志可使图形项的所有子图形项在图形项的范围内绘制，这有利于图形绘制和碰撞检测。与 ItemContains-ChildrenInShape 标志相比，该标志不是强制性的

在 QGraphicsItem 类中，可以通过重写 itemChange(change:QGraphicsItemChange, value:Any)函数设置当图形项发生改变时能够及时做出反应，参数 value 的值根据状态 change 决定，参数 change 的值为 QGraphicsItem. GraphicsItemChange 的枚举值。如果要使用 itemChange()函数处理几何位置改变的通知，则要通过 setFlag()方法给图形项设置 QGraphicsItem. ItemSendsGeometryChange 标志，而且不能在 itemChange()函数中直接改变几何位置，否则会陷入死循环。

QGraphicsItem. GraphicsItemChange 的枚举值见表 5-13。

表 5-13 QGraphicsItem. GraphicsItemChange 的枚举值

枚 举 值	说 明
QGraphicsItem. ItemEnabledChange	当图形项的激活状态(setEnable())即将改变时发送通知，itemChange()函数中的参数 value 表示新状态，value=True 表示图形项处于激活状态，value=False 表示图形项处于失效状态。原激活状态可使用 isEnabled()方法获得
QGraphicsItem. ItemEnabledHasChanged	当图形项的激活状态已经改变时发送通知，itemChange()函数中的参数 value 是新状态
QGraphicsItem. ItemPositonChange	当图形项的位置(setPos()、moveBy())即将改变时发送通知，参数 value 是相对于父图形项改变后的位置 QPointF，原位置可使用 pos()方法获得
QGraphicsItem. ItemPositionHasChanged	当图形项的位置已经改变时发送通知，参数 value 是相对于父图形项改变后的位置 QPointF，与 pos()方法获得的位置相同
QGraphicsItem. ItemTransformChange	当图形项的变换矩阵(setTransform())即将改变时发送通知，参数 value 是变换后的矩阵 QTransform，原变换矩阵可用 transform()方法获得
QGraphicsItem. ItemTransformHasChanged	当图形项的变换矩阵已经改变时发送通知，参数 value 是变换后的矩阵 QTransform，与 transform()方法获得的矩阵相同

续表

枚 举 值	说 明
QGraphicsItem. ItemRotationChange	当图形项即将产生旋转(setRotation())时发送通知,参数 value 是新的旋转角度,原旋转角度可用 rotation()方法获得
QGraphicsItem. ItemRotationHasChanged	当图形项已经产生旋转时发送通知,参数 value 是新的旋转角度,与 rotation()方法获得的旋转角度相同
QGraphicsItem. ItemScaleChange	当图形项即将进行缩放(setScale())时发送通知,参数 value 是新的缩放系数,原缩放系数可用 scale()方法获得
QGraphicsItem. ItemScaleHasChanged	当图形项已经进行了缩放时发送通知,参数 value 是新的缩放系数
QGraphicsItem. ItemTransformOriginPointChange	当图形项变换原点(setTransformOriginPoint())即将改变时发送通知,参数 value 是新的原点 QPointF,原变换原点可用 transformOriginPoint()方法获得
QGraphicsItem. ItemTransformOriginPointHasChanged	当图形项的原点已经改变时发送通知,参数 value 是新的原点 QPointF,原变换原点可用 transformOriginPoint()方法获得
QGraphicsItem. ItemSelectedChange	当图形项选中状态即将改变时(setSelected())发送通知,参数 value 是选中后的状态(True 或 False),原选中状态可用 isSelected()方法获得
QGraphicsItem. ItemSelectedHasChanged	当图形项的选中状态已经改变时发送通知,参数 value 是选中后的状态
QGraphicsItem. ItemVisibleChange	当图形项的可见性(setVisible())即将改变时发送通知,参数 value 是新状态,原可见性状态可用 isVisible()方法获得
QGraphicsItem. ItemVisibleHasChanged	当图形项的可见性已经改变时发送通知,参数 value 是新状态
QGraphicsItem. ItemParentChange	当图形项的父图形项(setParentItem())即将改变时发送通知,参数 value 是新的父图形项 QGraphicsItem,原父图形项可用 parentItem()方法获得
QGraphicsItem. ItemParentHasChanged	当图形项的父图形项已经改变时发送通知,参数 value 是新的父图形项
QGraphicsItem. ItemChildAddedChange	当图形项中即将添加子图形项时发送通知,参数 value 是新的子图形项,子图形项可能还没完全构建
QGraphicsItem. ItemChildRemoveChanged	当图形项中已经添加子图形项时发送通知,参数 value 是新的子图形项
QGraphicsItem. ItemSceneChange	当图形项即将加入场景(addItem())或即将从场景中(removeItem())移除时发送通知,参数 value 是新场景或 None(移除时),原场景可用 scene()方法获得
QGraphicsItem. ItemSceneHasChanged	当图形项已经加入场景中或即将从场景中移除时发送通知,参数 value 是新场景或 None(移除时)

续表

枚 举 值	说 明
QGraphicsItem. ItemCursorChange	当图形项的光标形状(setCursor())即将改变时发送通知,参数 value 是新光标 QCursor,原光标可用 cursor()方法获得
QGraphicsItem. ItemCursorHasChanged	当图形项的光标形状已经改变时发送通知,参数 value 是新光标 QCursor
QGraphicsItem. ItemToolTipChange	当图形项的提示信息(setToolTip())即将改变时发送通知,参数 value 是新提示信息,原提示信息可用 toolTip()方法获得
QGraphicsItem. ItemToolTipHasChanged	当图形项的提示信息已经改变时发送通知,参数 value 是新提示信息
QGraphicsItem. ItemFlagsChange	当图形项的标识(setFlags())即将改变时发送通知,参数 value 是新标识信息值
QGraphicsItem. ItemFlagsHaveChanged	当图形项的标识已经改变时发送通知,参数 value 是新标识信息值
QGraphicsItem. ItemZValueChange	当图形项的 z 值(setZValue())即将改变时发送通知,参数 value 是新的 z 值,原 z 值可用 zValue()方法获得
QGraphicsItem. ItemZValueHasChanged	当图形项的 z 值已经改变时发送通知,参数 value 是新的 z 值
QGraphicsItem. ItemOpacityChange	当图形项的不透明度(setOpacity())即将改变时发送通知,参数 value 是新的不透明度,原透明度可用 opacity()方法获得
QGraphicsItem. ItemOpacityHasChanged	当图形项的不透明度已经改变时发送通知,参数 value 是新的不透明度
QGraphicsItem. ItemScenePositionHasChanged	当图形项所在的场景位置已经发生改变时发送通知,参数 value 是新的场景位置,与 scenePos()方法获得的位置相同

图形项 QGraphicsItem 类的处理事件有 contextMenuEvent()、focusInEvent()、focusOutEvent()、hoverEnterEvent()、hoverMoveEvent()、hoverLeaveEvent()、inputMethodEvent()、keyPressEvent()、keyReleaseEvent()、mousePressEvent()、mouseMoveEvent()、mouseReleaseEvent()、mouseDoubleClickEvent()、dragEnterEvent()、dragLeaveEvent()、dragMoveEvent()、dropEvent()、wheelEvent()、sceneEvent(QEvent)。使用 installSceneEventFilter(QGraphicsItem)方法给事件添加过滤器。使用 sceneEventFilter(QGraphicsItem,QEvent)方法处理事件,并返回 bool 型数据。使用 removeSceneEventFilter(QGraphicsItem)方法移除事件过滤器。

【实例 5-3】 使用 Graphics/View 框架绘制图像,需包含两个自定义图形项。这两个自定义图形项存在父子关系,而且这两个图形项构成组合,代码如下:

```
# === 第 5 章 代码 demo3.py === #
import sys,math
from PySide6.QtWidgets import (QApplication,QWidget,QGraphicsScene,
QGraphicsView,QVBoxLayout,QGraphicsItem)
from PySide6.QtCore import Qt,QRectF,QPointF
from PySide6.QtGui import QPolygonF,QPainterPath

#自定义椭圆图形项
class ellipse(QGraphicsItem):
    def __init__(self,width,height,parent = None):
        super().__init__(parent)
        self.__width = width
        self.__height = height
     def boundingRect(self):
        return QRectF( - 5, - self.__height/2 - 20,
self.__width + 25,self.__height + 40)
    def paint(self, painter,option,widget):
        pen = painter.pen()
        pen.setWidth(3)
        painter.setPen(pen)
        #绘制椭圆
        painter.drawEllipse( - 10, - 1/2 * self.__height - 10,
self.__width,self.__height)
        #绘制文字
        font = painter.font()
        font.setPixelSize(20)
        painter.setFont(font)
        painter.drawText(QPointF(1/2 * self.__width,0),"椭圆的中心")

#自定义余弦曲线图形项
class cos(QGraphicsItem):
    def __init__(self,width,height,parent = None):
        super().__init__(parent)
        self.__width = width
        self.__height = height
    def boundingRect(self):
        return QRectF( - 5, - self.__height/2 - 20,self.__width + 25,
self.__height + 40)
    def paint(self,painter,option,widget):
        polygon_cos = QPolygonF()
        for i in range(360):
            x_value = i * self.__width/360
            cos_value = math.cos(i * math.pi/180) * ( - 1) * self.__height/2
            polygon_cos.append(QPointF(x_value,cos_value))
        pen = painter.pen()
        pen.setWidth(3)
        painter.setPen(pen)
        #绘制余弦曲线
        painter.drawPolyline(polygon_cos)

class Window(QWidget):
    def __init__(self,parent = None):
```

```
        super().__init__(parent)
        self.setGeometry(200,200,580,280)
        self.setWindowTitle("QGraphicsItem")
        #创建图像视图控件
        self.graphicsView = QGraphicsView()
        #设置布局
        vbox = QVBoxLayout(self)
        vbox.addWidget(self.graphicsView)
        w = 500                                          #正弦曲线的宽度
        h = 230                                          #正弦曲线的高度
        rectF = QRectF( - 10, - 10 - h/2,w,h)            #场景的范围
        #创建图像场景
        self.graphicsScene = QGraphicsScene(rectF)
        #图像视图设置图像场景
        self.graphicsView.setScene(self.graphicsScene)
        item1 = ellipse(w,h)                             #自定义椭圆图形项
        item2 = cos(w,h)                                 #自定义正弦曲线图形项
        item2.setParentItem(item1)                       #设置图形项的父子关系
        self.graphicsScene.addItem(item1)                #添加自定义的图形项
        rectangle = self.graphicsScene.addRect(rectF)    #添加矩形边框
        #创建组合
        group = self.graphicsScene.createItemGroup([item1,rectangle])
        #设置组合可移动
        group.setFlag(QGraphicsItem.ItemIsMovable)

if __name__ == "__main__":
    app = QApplication(sys.argv)
    win = Window()
    win.show()
    sys.exit(app.exec())
```

运行结果如图 5-6 所示。

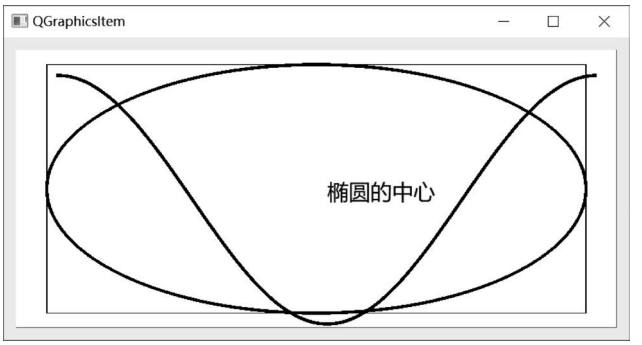

图 5-6　代码 demo3.py 的运行结果

5.2.4　标准图形项类

在 Graphics/View 框架中，不仅可以自定义图形项，也可以使用标准图形项。标准图形项类有 QGraphicsLineItem、QGraphicsRectItem、QGraphicsPolygonItem、QGraphicsEllipseItem、

QGraphicsPathItem、QGraphicsPixmapItem、QGraphicsSimpleTextItem 、QGraphicsTextItem 。这些类都继承自 QGraphicsItem 类,使用这些类可以创建标准图形项,然后使用图形场景类 QGraphicsScene 的 addItem()方法向图像场景中添加标准图形项。

8 个标准图形项类的继承关系如图 5-7 和图 5-8 所示。

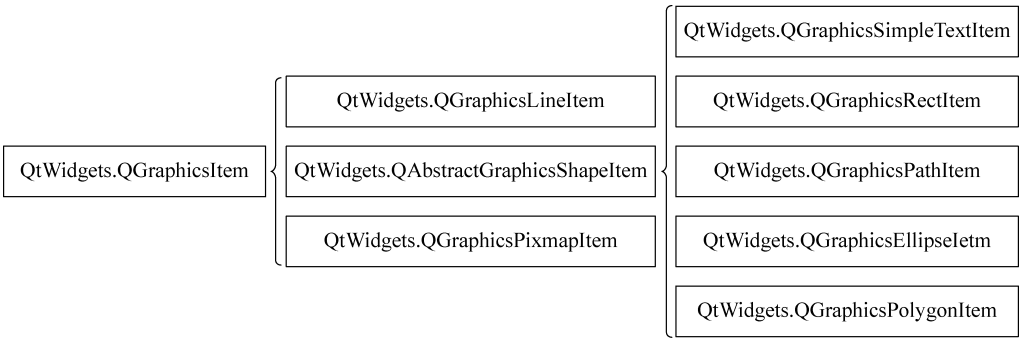

图 5-7 8 个标准图形项类的继承关系

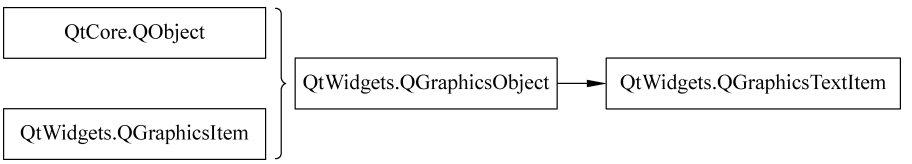

图 5-8 QGraphicsTextItem 类的继承关系

1. 直线图形项类 QGraphicsLineItem

使用 QGraphicsLineItem 类可以创建直线图形项,其构造函数如下:

```
QGraphicsLineItem(parent:QGraphicsItem = None)
QGraphicsLineItem(line:Union[QLineF,QLine],parent:QGraphicsItem = None)
QGraphicsLineItem(x1:float,y1:float,x2:float,y2:float,parent:QGraphicsItem = None)
```

其中,parent 表示 QGraphicsItem 类及其子类创建的实例对象。

QGraphicsLineItem 类的常用方法见表 5-14。

表 5-14 QGraphicsLineItem 类的常用方法

方法及参数类型	说 明	返回值的类型
setLine(line:Union[QLineF,QLine])	设置线段	None
setLine(x1:float,y1:float,x2:float,y2:float)	设置线段	None
setPen(pen:Union[QPen,QPenStyle,QColor])	设置钢笔	None
line()	获取线段	QLineF
pen()	获取钢笔	QPen

2. 矩形图形项类 QGraphicsRectItem

使用 QGraphicsRectItem 类可以创建矩形图形项,其构造函数如下:

```
QGraphicsRectItem(parent:QGraphicsItem = None)
QGraphicsRectItem(rect:Union[QRectF,QRect],parent:QGraphicsItem = None)
QGraphicsRectItem(x:float,y:float,w:float,h:float,parent:QGraphicsItem = None)
```

其中，parent 表示 QGraphicsItem 类及其子类创建的实例对象。

QGraphicsRectItem 类的常用方法见表 5-15。

表 5-15 QGraphicsRectItem 类的常用方法

方法及参数类型	说 明	返回值的类型
setRect(rect:Union[QRectF,QRect])	设置矩形	None
setRect(x:float,y:float,w:float,h:float)	设置矩形	None
rect()	获取矩形	QRectF
setPen(pen:Union[QPen,Qt.PenStyle,QColor])	设置钢笔	None
pen()	获取钢笔	QPen
setBrush(brush:Union[QBrush,Qt.BrushStyle,QColor,Qt.GlobalColor,QGradient,QImage,QPixmap])	设置画刷	None
brush()	获取画刷	QBrush

3. 多边形图形项类 QGraphicsPolygonItem

使用 QGraphicsPolygonItem 类可以创建多边形图形项，其构造函数如下：

```
QGraphicsPolygonItem(parent:QGraphicsItem = None)
QGraphicsPolygonItem(polygon:Union[QPolygonF,QPolygon,Sequence[QPointF],QRectF],parent:
QGraphicsItem = None)
```

其中，parent 表示 QGraphicsItem 类及其子类创建的实例对象。

QGraphicsPolygonItem 类的常用方法见表 5-16。

表 5-16 QGraphicsPolygonItem 类的常用方法

方法及参数类型	说 明	返回值的类型
setPolygon(polygon:Union[QPolygonF,QPolygon,Sequence[QPointF],QRectF])	设置多边形	None
polygon()	获取多边形	QPolygonF
setFillRule(Qt.FillRule=Qt.OddEventFill)	设置填充规则	None
fillRule()	获取填充规则	Qt.FillRule
setPen(pen:Union[QPen,Qt.PenStyle,QColor])	设置钢笔	None
pen()	获取钢笔	QPen
setBrush(brush:Union[QBrush,Qt.BrushStyle,QColor,Qt.GlobalColor,QGradient,QImage,QPixmap])	设置画刷	None
brush()	获取画刷	QBrush

4. 椭圆图形项类 QGraphicsEllipseItem

使用 QGraphicsEllipseItem 类可以创建椭圆图形项，其构造函数如下：

```
QGraphicsEllipseItem(parent:QGraphicsItem = None)
QGraphicsEllipseItem(rect:Union[QRectF,QRect],parent:QGraphicsItem = None)
QGraphicsEllipseItem(x:float,y:float,w:float,h:float,parent:QGraphicsItem = None)
```

其中,parent 表示 QGraphicsItem 类及其子类创建的实例对象。

QGraphicsEllipseItem 类的常用方法见表 5-17。

表 5-17 QGraphicsEllipseItem 类的常用方法

方法及参数类型	说 明	返回值的类型
setRect(rect:Union[QRectF,QRect])	设置椭圆的范围	None
setRect(x:float,y:float,w:float,h:float)	设置椭圆的范围	None
rect()	获取椭圆的范围	QRectF
setSpanAngle(angle:int)	设置跨度角度	None
spanAngle()	获取跨度角度	int
setStartAngle(angle:int)	设置起始角度	None
startAngle()	获取起始角度	int
setPen(pen:Union[QPen,Qt.PenStyle,QColor])	设置钢笔	None
pen()	获取钢笔	QPen
setBrush(brush:Union[QBrush,Qt.BrushStyle,QColor,Qt.GlobalColor,QGradient,QImage,QPixmap])	设置画刷	None
brush()	获取画刷	QBrush

5. 路径图形项类 QGraphicsPathItem

使用 QGraphicsPathItem 类可以创建路径图形项,其构造函数如下:

```
QGraphicsPathItem(parent:QGraphicsItem = None)
QGraphicsPathItem(path:QPainterPath,parent:QGraphicsItem = None)
```

其中,parent 表示 QGraphicsItem 类及其子类创建的实例对象。

QGraphicsPathItem 类的常用方法见表 5-18。

表 5-18 QGraphicsPathItem 类的常用方法

方法及参数类型	说 明	返回值的类型
setPath(path:QPainterPath)	设置路径	None
path()	获取路径	QPainterPath
setPen(pen:Union[QPen,Qt.PenStyle,QColor])	设置钢笔	None
pen()	获取钢笔	QPen
setBrush(brush:Union[QBrush,Qt.BrushStyle,QColor,Qt.GlobalColor,QGradient,QImage,QPixmap])	设置画刷	None
brush()	获取画刷	QBrush

6. 图像图形项类 QGraphicsPixmapItem

使用 QGraphicsPixmapItem 类可以创建路径图形项,其构造函数如下:

```
QGraphicsPixmapItem(parent:QGraphicsItem = None)
QGraphicsPixmapItem(pixmap:Union[QPixmap,QImage,str],parent:QGraphicsItem = None)
```

其中，parent 表示 QGraphicsItem 类及其子类创建的实例对象。

QGraphicsPixmapItem 类的常用方法见表 5-19。

表 5-19 QGraphicsPixmapItem 类的常用方法

方法及参数类型	说 明	返回值的类型
setOffset(offset:Union[QPointF,QPoint,QPainterPath.Element])	设置图像左上角的坐标	None
setOffset(x:float,y:float)	设置图像左上角的坐标	None
offset()	获取图像左上角的坐标	QPointF
setPixmap(pixmap:Union[QPixmap,QImage,str])	设置图像	None
pixmap()	获取图像	QPixmap
setShapeMode(QGraphicsPixmapItem.ShapeMode)	设置计算形状的方法	None
setTransformationMode(Qt.TransformationMode)	设置图像的变换模式	None
shapeMode()	获取计算形状的方法	ShapeMode
transformationMode()	获取图像的变换模式	TransformationMode

在表 5-19 中，QGraphicsPixmapItem.ShapeMode 的枚举值为 QGraphicsPixmapItem.MaskShape(通过调用 QPixmap.mask()计算形状)、QGraphicsPixmapItem.BoundingRectShape(通过轮廓计算形状)、QGraphicsPixmapItem.HeurisiticMaskShape(通过调用 QPixmap.createHeuristicMask()方法确定形状)。

Qt.TransformatiomMode 的枚举值为 Qt.FastTransformation(快速变换)、Qt.SmoothTransformation(光滑变换)。

7. 纯文本图形项类 QGraphicsSimpleTextItem

使用 QGraphicsSimpleTextItem 类可以创建纯文本图形项，其构造函数如下：

```
QGraphicsSimpleTextItem(parent:QGraphicsItem = None)
QGraphicsSimpleTextItem(text:str,parent:QGraphicsItem = None)
```

其中，parent 表示 QGraphicsItem 类及其子类创建的实例对象。

QGraphicsSimpleTextItem 类的常用方法见表 5-20。

表 5-20 QGraphicsSimpleTextItem 类的常用方法

方法及参数类型	说 明	返回值的类型
setText(str)	设置文本	None
text()	获取文本	str
setFont(font)	设置字体	None
font()	获取字体	QFont
setBrush(brush:Union[QBrush,Qt.BrushStyle,QColor,Qt.GlobalColor,QGradient,QImage,QPixmap])	设置文本的填充色	None
setPen(pen:Union[QPen,Qt.PenStyle,QColor])	设置钢笔	None

8. 文本图形项类 QGraphicsTextItem

使用 QGraphicsTextItem 类可以创建具有格式、可编辑的文本图形项,其构造函数如下:

```
QGraphicsTextItem(parent:QGraphicsItem = None)
QGraphicsTextItem(text:str,parent:QGraphicsItem = None)
```

其中,parent 表示 QGraphicsItem 类及其子类创建的实例对象。

QGraphicsTextItem 类的常用方法见表 5-21。

表 5-21 QGraphicsTextItem 类的常用方法

方法及参数类型	说 明	返回值的类型
adjustSize()	调整到合适的尺寸	None
openExternLinks()	获取是否打开外部链接	bool
setDefaultTextColor(Union[QColor, Qt.GlobalColor, QGradient])	设置文本的默认颜色	None
setDocument(QTextDocument)	设置文档	None
setFont(QFont)	设置字体	None
setHtml(str)	设置 HTML 格式文本	None
toHtml()	将文本转换为 HTML 格式文本	str
setOpenExternalLinks(bool)	设置是否打开外部链接	None
setPlainText(str)	设置纯文本	None
toPlainText()	转换为纯文本	str
setTabChangesFocus(bool)	是否设置 Tab 键可移动焦点	None
setTextCursor()	设置文本光标	None
setTextInteractionFlags(Qt.TextInteractionFlag)	设置标志,以确定文本项如何响应用户的输入	None

在表 5-21 中,Qt.TextInteractionFlag 的枚举值为 Qt.NoTextInteraction、Qt.SelectableByMouse、Qt.TextSelectionByKeyboard、Qt.LinksAccessibleByMouse、Qt.LinksAccessibleByKeyboard、Qt.TextEditable、Qt.TextEditorInteraction(表示 Qt.SelectableByMouse| Qt.TextSelectionByKeyboard | Qt.TextEditable)、Qt.TextBrowerInteraction(表示 Qt.SelectableByMouse| Qt.LinksAccessibleByMouse| LinksAccessibleByKeyboard)。

与其他标准图形项类不同,QGraphicsTextItem 类具有鼠标事件和键盘事件。QGraphicsTextItem 类的信号见表 5-22。

表 5-22 QGraphicsTextItem 类的信号

信号及参数类型	说 明
linkActivated(link)	当单击超链接时发送信号
linkHovered(link)	当光标在超链接上悬停时发送信号

【实例 5-4】 创建一个窗口，该窗口包含 1 个菜单栏、1 个工具栏和 7 个工具按钮。使用该窗口可绘制直线、矩形、椭圆、圆，并可以停止绘图、删除指定图形项、清空所有图形项，代码如下：

```
# === 第 5 章 代码 demo4.py === #
import sys,math
from PySide6.QtWidgets import (QApplication,QMainWindow,QGraphicsScene,
QGraphicsView,QGraphicsItem)
from PySide6.QtCore import Qt,Signal,QPoint,QRectF,QPointF,QLineF
from PySide6.QtGui import QPolygonF

#创建视图控件的子类
class myGraphicsView(QGraphicsView):
    press_point = Signal(QPointF)     #自定义信号,参数为鼠标被按下时鼠标在视图中的位置
    move_point = Signal(QPointF)      #自定义信号,参数为移动鼠标时鼠标在视图中的位置
    release_point = Signal(QPointF)   #自定义信号,参数为鼠标被释放时鼠标在视图中的位置
    def __init__(self,parent = None):
        super().__init__(parent)
    #按下鼠标按键事件
    def mousePressEvent(self,event):
        self.press_point.emit(event.position())       #发送信号,参数是鼠标位置
        super().mousePressEvent(event)
    #鼠标移动事件
    def mouseMoveEvent(self,event):                   #鼠标移动事件
        self.move_point.emit(event.position())        #发送信号,参数是鼠标位置
        super().mouseMoveEvent(event)
    #鼠标按键被释放
    def mouseReleaseEvent(self,event):
        self.release_point.emit(event.position())
        super().mouseReleaseEvent(event)


class Window(QMainWindow):
    def __init__(self,parent = None):
        super().__init__(parent)
        self.resize(580,280)
        self.setWindowTitle("自定义图形项类")
        #shape 用于记录哪个绘图按钮被选中
        self.shape = {'直线':False,'矩形':False,'椭圆':False,'圆':False}
        self.__temp = None            #用于指向鼠标移动时产生的临时图形项
        #创建图像视图控件
        self.graphicsView = myGraphicsView()
        self.setCentralWidget(self.graphicsView)
        rectF = QRectF(self.width()/2,self.height()/2,self.width(),self.height())
        #创建图像场景
        self.graphicsScene = QGraphicsScene(rectF)
        self.graphicsView.setViewportUpdateMode(QGraphicsView.FullViewportUpdate)
        #图像视图设置图像场景
        self.graphicsView.setScene(self.graphicsScene)
```

```
        #使用信号/槽
        self.graphicsView.press_point.connect(self.press_position)
        self.graphicsView.move_point.connect(self.move_position)
        self.graphicsView.release_point.connect(self.release_position)
        #创建菜单栏
        self.menubar = self.menuBar()
        #创建菜单
        self.draw = self.menubar.addMenu('绘图')
        #给菜单添加动作
        action_line = self.draw.addAction('直线')
        action_rect = self.draw.addAction('矩形')
        action_ellipse = self.draw.addAction('椭圆')
        action_circle = self.draw.addAction('圆')
        self.draw.addSeparator()                                    #添加分隔符
        action_stop = self.draw.addAction('停止')
        action_delete = self.draw.addAction("删除")
        action_clear = self.draw.addAction("清空")
        #使用信号/槽
        action_line.triggered.connect(self.line_triggered)
        action_rect.triggered.connect(self.rect_triggered)
        action_ellipse.triggered.connect(self.ellipse_triggered)
        action_circle.triggered.connect(self.cirle_triggered)
        action_stop.triggered.connect(self.stop_triggered)
        action_delete.triggered.connect(self.delete_triggered)
        action_clear.triggered.connect(self.graphicsScene.clear)
        action_clear.triggered.connect(self.graphicsScene.update)
        #创建工具栏
        self.toolbar_draw = self.addToolBar("绘图")
        self.toolbar_draw.addAction(action_line)
        self.toolbar_draw.addAction(action_rect)
        self.toolbar_draw.addAction(action_ellipse)
        self.toolbar_draw.addAction(action_circle)
        self.toolbar_draw.addSeparator()
        self.toolbar_draw.addAction(action_stop)
        self.toolbar_draw.addSeparator()
        self.toolbar_draw.addAction(action_delete)
        self.toolbar_draw.addAction(action_clear)
    #鼠标按下
    def press_position(self,pointF):
        point = pointF.toPoint()
        self.__pressPos = self.graphicsView.mapToScene(point)       #映射成场景坐标
    #鼠标移动
    def move_position(self,pointF):
        point = pointF.toPoint()
        self.__movePos = self.graphicsView.mapToScene(point)
        self.move_draw(self.__pressPos,self.__movePos)              #调用绘图函数
    #鼠标释放
    def release_position(self,pointF):
        point = pointF.toPoint()
        if self.__temp:
            self.__temp.setFlags(QGraphicsItem.ItemIsSelectable
|QGraphicsItem.ItemIsFocusable)
```

```
            self.__temp = None
        rect = self.graphicsScene.itemsBoundingRect()
        if rect.width()> self.width() or rect.height()> self.height():
            self.graphicsScene.setSceneRect(rect)
    #绘制直线
    def line_triggered(self):
        self.shape = {'直线':True,'矩形':False,'椭圆':False,'圆':False}
    #绘制矩形
    def rect_triggered(self):
        self.shape = {'直线':False,'矩形':True,'椭圆':False,'圆':False}
    #绘制椭圆
    def ellipse_triggered(self):
        self.shape = {'直线':False,'矩形':False,'椭圆':True,'圆':False}
    #绘制圆
    def circle_triggered(self):
        self.shape = {'直线':False,'矩形':False,'椭圆':False,'圆':True}
    #停止绘制
    def stop_triggered(self):
        self.shape = {'直线':False,'矩形':False,'椭圆':False,'圆':False}
    #清空图形项
    def delete_triggered(self):
        if len(self.graphicsScene.selectedItems()):
            for i in self.graphicsScene.selectedItems():
                self.graphicsScene.removeItem(i)
    #当鼠标移动时绘制图形项
    def move_draw(self,p1,p2):
        x1 = min(p1.x(), p2.x())
        y1 = min(p1.y(), p2.y())
        x2 = max(p1.x(), p2.x())
        y2 = max(p1.y(), p2.y())
        rectF = QRectF(QPointF(x1, y1), QPointF(x2, y2))#鼠标按下点与移动点的矩形区域
        if self.__temp:     #在鼠标移动过程中,如果变量已经指向图形项,则需要把图形项移除
            self.graphicsScene.removeItem(self.__temp)
        if self.shape['直线']:
            self.__temp = self.graphicsScene.addLine(QLineF(p1,p2))     #添加直线
        if self.shape['矩形']:
            self.__temp = self.graphicsScene.addRect(rectF)             #添加矩形
        if self.shape['椭圆']:
            self.__temp = self.graphicsScene.addEllipse(rectF)          #添加椭圆
        if self.shape['圆']:
            r = math.sqrt((p1.x() - p2.x()) ** 2 + (p1.y() - p2.y()) ** 2)
            pointF_1 = QPointF(p1.x() - r,p1.y() - r)
            pointF_2 = QPointF(p1.x() + r,p1.y() + r)
            self.__temp = self.graphicsScene.addEllipse(
QRectF(pointF_1,pointF_2))                                              #添加圆


if __name__ == '__main__':
    app = QApplication(sys.argv)
    win = Window()
    win.show()
    sys.exit(app.exec())
```

运行结果如图 5-9 所示。

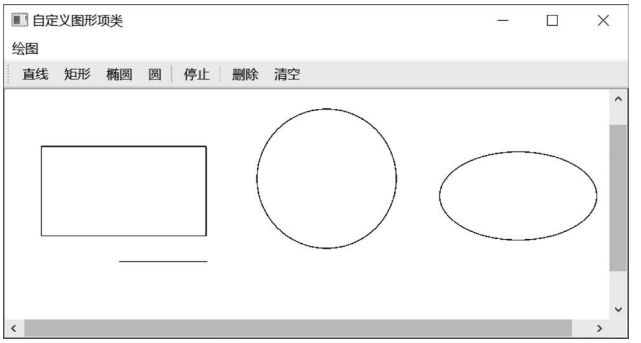

图 5-9 代码 demo4.py 的运行结果

5.3 代理控件和图形控件

在 Graphics/View 框架中,不仅可以向图像场景中添加图形项,也可以添加控件、对话框,而且可以在图像场景中对控件进行布局管理。


10min

5.3.1 代理控件类 QGraphicsProxyWidget

在图像场景类 QGraphicsScene 中,可通过 addWidget(QWidget,wFlags:Qt.WindowFlags)方法向图像场景中添加控件或窗口,并返回代理控件对象 QGraphicsProxyWidget。

使用 QGraphicsProxyWidget 类可以创建代理控件。可以使用代理控件的 setWidget(QWidget)方法设置控件或窗口,然后使用图像场景类 QGraphicsScene 的 addItem(QGraphicsProxyWidget)方法向图像场景中添加代理控件。

QGraphicsProxyWidget 类的继承关系如图 5-10 所示。

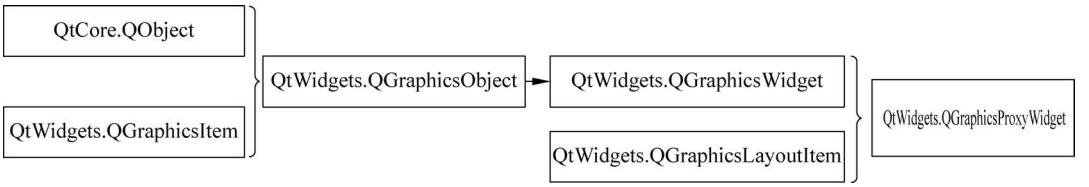

图 5-10 QGraphicsProxyWidget 类的继承关系

QGraphicsProxyWidget 类位于 PySide6 的 QtWidgets 子模块下,其构造函数如下:

```
QGraphicsProxyWidget(parent:QGraphicsItem = None,wFlags:Qt.WindowFlags)
```

其中,parent 表示 QGraphicsItem 类及其子类创建的实例对象。

QGraphicsProxyWidget 类的常用方法见表 5-23。

表 5-23 QGraphicsProxyWidget 类的常用方法

方法及参数类型	说 明	返回值的类型
setWidget(QWidget)	添加控件	None
widget()	获取控件	QWidget
createProxyForChildWidget(QWidget)	为代理控件中的控件创建代理控件	None
subWidgetRect()	获取代理控件中控件的范围	QRectF

在 Graphics/View 框架中,代理控件与其内部的控件保持同步的状态,例如激活状态、可见性、字体、调色板、光标形状、窗口标题、几何尺寸、布局方向。

【实例 5-5】 自定义一个窗口类,该窗口包含一个标签控件、一个按钮控件,单击该按钮可打开并显示图像文件。将该窗口类创建的窗口控件显示在 Graphics/View 框架下的图像场景中,而且要使用错切变换,代码如下:

```
# === 第 5 章 代码 demo5.py === #
import sys,os
from PySide6.QtWidgets import (QApplication,QWidget,QVBoxLayout,
QGraphicsProxyWidget,QGraphicsScene,QGraphicsView,QPushButton,QFileDialog,QLabel)
from PySide6.QtGui import QTransform,QPixmap
from PySide6.QtCore import Qt

#创建一个可以显示图像的窗口类
class PixmapWidget(QWidget):
    def __init__(self,parent = None):
        super().__init__(parent)
        self.resize(580,280)
        self.setWindowTitle("代理控件内的窗口")
        #创建标签
        self.label = QLabel()
        #创建按钮
        self.button = QPushButton("选择图像文件")
        #设置布局
        vbox = QVBoxLayout(self)                          #布局
        vbox.addWidget(self.label)
        vbox.addWidget(self.button)
        #使用信号/槽
        self.button.clicked.connect(self.button_clicked)
    def button_clicked(self):
        fileName, fil = QFileDialog.getOpenFileName(self, caption = "打开图像文件",filter =
"图像(*.png *.bmp *.jpg *.jpeg)")
        if os.path.exists(fileName) == False:
            return
        pix = QPixmap(fileName)
        pix = pix.scaled(580,280)                         #缩放图像文件
        self.label.setPixmap(pix)

class Window(QWidget):
    def __init__(self,parent = None):
        super().__init__(parent)
        self.setWindowTitle("代理控件")
```

```
        pix = PixmapWidget()                                          #创建自定义窗口
        view = QGraphicsView()                                        #创建图像视图控件
        scene = QGraphicsScene()                                      #创建图像场景控件
        view.setScene(scene)                                          #在图像视图中设置场景
        proxy = QGraphicsProxyWidget(None,Qt.Window)                  #创建代理控件
        proxy.setWidget(pix)                                          #代理控件设置控件
        proxy.setTransform(QTransform().shear(-0.8,-0.1))             #错切变换
        scene.addItem(proxy)                                          #在场景中添加代理控件
        vbox = QVBoxLayout(self)                                      #设置布局
        vbox.addWidget(view)

if __name__ == '__main__':
    app = QApplication(sys.argv)
    win = Window()
    win.show()
    sys.exit(app.exec())
```

运行结果如图 5-11 所示。

图 5-11　代码 demo5.py 的运行结果

5.3.2　图形控件类 QGraphicsWidget

在 PySide6 中,使用 QGraphicsWidget 类创建图形控件。由于 QGraphicsWidget 类继承自 QGraphicsItem 类,因此图形控件可直接添加到图像场景中。QGraphicsWidget 类的继承关系如图 5-10 所示。

QGraphicsWidget 类是所有图形控件类的基类,其子类包括 QtWidgets.QGraphicsProxyWidget、QtCharts.QChart、QtCharts.QLegend、QtCharts.QPolarChart。在 QGraphicsWidget 类创建的图形控件中,可以添加代理控件和布局,因此图形控件也可以作为图像场景的容器使用。QGraphicsWidget 类的构造函数如下:

```
QGraphicsWidget(parent:QGraphicsItem = None,wFlag:Qt.WindowFlags = Default(Qt.WindowFlags))
```

其中,parent 表示 QGraphicsItem 类及其子类创建的实例对象。

QGraphicsWidget类与Widget类进行对比，既有相同点，也有不同点。QGraphicsWidget类的常用方法见表5-24。

表5-24 QGraphicsWidget类的常用方法

方法及参数类型	说 明	返回值的类型
[static]setTabOrder(first:QGraphicsWidget,second:QGraphicsWidget)	设置按Tab键获取焦点的顺序	None
[slot]close()	关闭窗口，若成功，则返回值为True	bool
setAttribute(attribute:Qt.WidgetAttribute,on:bool=True)	设置属性	None
testAttribute(attribute:Qt.WidgetAttribute)	测试是否设置了某种属性	bool
itemChange(change:QGraphicsItem.GraphicsItemChange,value:Any)	重写该函数，作为信号使用	None
paint(painter:QPainter,option:QStyleOptionGraphicsItem,widget:QWidget=None)	重写该函数，绘制图形	None
boundingRect()	重写该函数，获取边界矩形	QRectF
shape()	重写该函数，获取路径对象	QPainterPath
setLayout(layout:QGraphicsLayout)	设置布局	None
layout()	获取布局	QGraphicsLayout
setLayoutDirection(direction:Qt.LayoutDirection)	设置布局方向	None
setAutoFillBackground(enabled:bool)	设置是否自动填充背景	None
setContentsMargins(margins:Union[QMraginF,QMargins])	设置窗口内的控件到边框的最小距离	None
setContentsMargins(left:float,top:float,right:float,bottom:float)	设置窗口内的控件到边框的最小距离	None
setFocusPolicy(policy:Qt.FocusPolicy)	设置获取焦点的策略	None
setFont(font:Union[QFont,str,Sequence[str]])	设置字体	None
setGeometry(x:float,y:float,w:float,h:float)	设置位置，以及宽和高	None
setGeometry(rect:Union[QRectF,QRect])	设置位置，以及宽和高	None
setPalette(palette:Union[QPalette,Qt.GlobalColor,QColor])	设置调色板	None
setStyle(style:QStyle)	设置风格	None
setWindowFlags(wFlags:Qt.WindowFlags)	设置窗口标识	None
setWindowFrameMargins(Union[QMarginF,QMargins])	设置边框距	None
setWindowFrameMargins(float,float,float,float)	设置边框距	None
setWindowTitle(title:str)	设置窗口标题	None
rect()	获取图形控件的窗口范围	QRectF
resize(QSizeF)	调整窗口的宽和高	None
resize(float,float)	调整窗口的宽和高	None
size()	获取窗口的宽和高	QSizeF
focusWidget()	获取焦点控件	QGraphicsWidget

续表

方法及参数类型	说　明	返回值的类型
isActiveWindow()	获取是否为活跃控件	bool
updateGeometry()	刷新图形控件	None
addAction(QAction)	向图形控件中添加动作	None
addActions(Sequence[QAction])	向图形控件中添加动作	None
insertActions(before:QAction,actions:Sequence[QAction])	向图形控件中插入动作,图形控件的动作可以作为右键菜单使用	None
insertAction(before:QAction,action:QAction)	向图形控件中插入动作,图形控件的动作可以作为右键菜单使用	None
removeAction(action:QAction)	移除指定动作	None

QGraphicsWidget 类的信号见表 5-25。

表 5-25　QGraphicsWidget 类的信号

信号及参数类型	说　明
geometryChanged()	当控件的几何宽和高发生改变时发送信号
layoutChanged()	当控件的布局发生改变时发送信号
childrenChanged()	当子控件的激活状态发生改变时发送信号
enabledChanged()	当控件的激活状态发生改变时发送信号
opacityChanged()	当控件的不透明度发生改变时发送信号
parentChanged()	当控件的父窗口发生改变时发送信号
rotationChanged()	当控件的旋转角度发生改变时发送信号
scaleChanged()	当控件的缩放发生改变时发送信号
visibleChanged()	当控件的可见性发生改变时发送信号
xChanged()	当控件的 x 坐标发生改变时发送信号
yChanged()	当控件的 y 坐标发生改变时发送信号
zChanged()	当控件的 z 坐标发生改变时发送信号

5.3.3　图形控件布局类

在 PySide6 中,可以设置图形控件的布局。图形控件的布局类有 3 种,分别为 QGraphicsLinearLayout、QGraphicsGridLayout、QGraphicsAnchorLayout,这 3 个类的继承关系如图 5-12 所示。

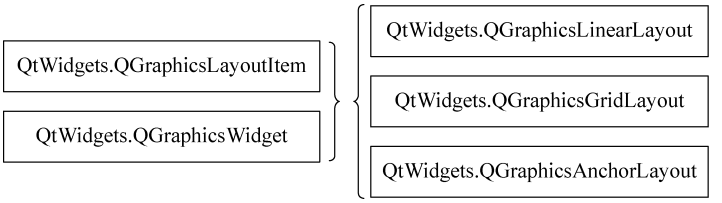

图 5-12　图形控件布局类的继承关系

1. 线性布局类 QGraphicsLinearLayout

使用 QGraphicsLinearLayout 类可以创建线性布局对象，线性布局对象内的图形控件呈线性分布，类似于垂直布局（QHLayoutBox）或水平布局（QVLayoutBox）。QGraphicsLinearLayout 类的构造函数如下：

```
QGraphicsLinearLayout(parent:QGraphicsLayoutItem = None)
QGraphicsLinearLayout(orientation:Qt.Orientation,parent:QGraphicsLayoutItem = None)
```

其中，parent 表示 QGraphicsLayoutItem 类及其子类创建的实例对象；orientation 表示线性布局的方向，参数值为 Qt.Orientation 的枚举值：Qt.Horizontal（水平方向）、Qt.Vertical（垂直方向）。

QGraphicsLinearLayout 类的常用方法见表 5-26。

表 5-26 QGraphicsLinearLayout 类的常用方法

方法及参数类型	说 明	返回值的类型
addItem(item:QGraphicsLayoutItem)	添加图形控件、代理控件、布局	None
insertItem(index:int,item:QGraphicsLayoutItem)	根据索引插入图形控件、布局	None
addStretch(stretch:int=1)	在末尾添加拉伸系数	None
insertStretch(index:int,stretch:int=1)	根据索引插入拉伸系数	None
count()	获取图形控件和布局的个数	int
setAlignment(QGraphicsLayoutItem,Qt.Alignment)	设置图形控件的对齐方式	None
setGeometry(rect:Union[QRectF,QRect])	设置布局的位置，以及宽和高	None
setItemSpacing(index:int,spacing:float)	根据索引设置间距	None
setOrientation(Qt.Orientation)	设置布局方向	None
setSpacing(spacing:float)	设置图形控件之间的间距	None
setStretchFactor(item: QGraphicsLayoutItem,stretch:int)	设置图形控件的拉伸系数	None
stretchFactor(item: QGraphicsLayoutItem)	获取图形控件的拉伸系数	int
itemAt(index:int)	根据索引获取图形控件或布局	QGraphicsWidget
removeAt(index:int)	根据索引移除图形控件或布局	None
removeItem(item:QGraphicsLayoutItem)	移除指定的图形控件或布局	None

2. 栅格布局类 QGraphicsGridLayout

栅格布局也称为网格布局，由多行多列构成。使用 QGraphicsGridLayout 类可以创建栅格布局对象，栅格布局对象中的图形控件可以占用一个单元格，也可以占用多行多列。QGraphicsGridLayout 类的构造函数如下：

```
QGraphicsGridLayout(parent:QGraphicsLayoutItem = None)
```

其中，parent 表示 QGraphicsLayoutItem 类及其子类创建的实例对象。

QGraphicsGridLayout 类的常用方法见表 5-27。

表 5-27 QGraphicsGridLayout 类的常用方法

方法及参数类型	说 明	返回值的类型
addItem(item:QGraphicsLayoutItem,row:int,column:int,alignment:Qt.Alignment=Default(Qt.Alignment))	在指定的位置添加图形控件	None
addItem(item:QGraphicsLayoutItem,row:int,column:int,rowSpan:int,columnSpan:int,alignment:Qt.Alignment)	添加图形控件,可占据多行多列	None
rowCount()	获取行数	int
columnCount()	获取列数	int
count()	获取图形控件和布局的个数	int
itemAt(row:int,column:int)	获取指定行、列处的图形控件或布局	QGraphicsWidget
itemAt(index:int)	根据索引获取图形控件或布局	QGraphicsWidget
removeAt(index:int)	根据索引移除图形控件或布局	None
removeItem(QGraphicsLayoutItem)	移除指定的图形控件或布局	None
setGeometry(rect:Union[QRectF,QRect])	设置位置,以及宽和高	None
setAlignment(QGraphicsLayoutItem,Qt.Alignment)	设置指定控件的对齐方法	None
setRowAlignment(row:int,alignment:Qt.Alignment)	设置行对齐方式	None
setColumnAlignment(row:int,alignment:Qt.Alignment)	设置列对齐方式	None
setRowFixedHeight(row:int,height:float)	设置行的固定高度	None
setRowMaximumHeight(row:int,height:float)	设置行的最大高度	None
setRowMinimumHeight(row:int,height:float)	设置行的最小高度	None
setRowPreferedHeight (row:int,height:float)	设置指定行的高度	None
setRowSpacing(row:int,spacing:float)	设置指定行的间距	None
setRowStretchFactor(row:int,stretch:int)	设置指定行的拉伸系数	None
setColumnFixedWidth(column:int,width:float)	设置列的固定宽度	None
setColumnMaximumWidth(column:int,width:float)	设置列的最大宽度	None
setColumnMinimumWidth(column:int,width:float)	设置列的最小宽度	None
setColumnPreferedWidth(column:int,width:float)	设置指定列的宽度	None
setColumnSpacing(column:int,spacing:float)	设置指定列的间距	None
setColumnStretchFactor(column:int,stretch:int)	设置指定列的拉伸系数	None
setSpacing(spacing:float)	设置行、列之间的间距	None
setHorizontalSpacing(spacing:float)	设置水平间隙	None
setVerticalSpacing(spacing:float)	设置竖直间隙	None

3. 锚点布局类 QGraphicsAnchorLayout

使用 QGraphicsAnchorLayout 类可以创建锚点布局对象。使用锚点布局可以设置两个图形控件之间的相对位置,例如两条边对齐、两个点对齐。QGraphicsAnchorLayout 类的构造函数如下:

```
QGraphicsAnchorLayout(parent:QGraphicsLayoutItem = None)
```

其中，parent 表示 QGraphicsLayoutItem 类及其子类创建的实例对象。

QGraphicsAnchorLayout 类的常用方法见表 5-28。

表 5-28 QGraphicsAnchorLayout 类的常用方法

方法及参数类型	说 明	返回值的类型
addAnchor(firstItem:QGraphicsLayoutItem,firstEdge:Qt. AnchorPoint, secondItem: QGraphicsLayoutItem, secondEdge:Qt. AnchorPoint)	将第 1 个图形控件的某条边和第 2 个图形控件的某条边对齐	None
addAnchors(firstItem:QGraphicsLayoutItem,secondItem:QGraphicsLayoutItem,orientations:Qt. Orientations)	设置两个图形控件在某条方向上宽和高相等	None
addCornerAnchors(firstItem:QGraphicsLayoutItem, firstEdge:Qt. AnchorPoint,secondItem: QGraphicsLayoutItem,secondEdge:Qt. AnchorPoint)	将第 1 个图形控件的某个角点和第 2 个图形控件的某个角点对齐	None
horizontalSpacing()	获取水平间距	float
setHorizontalSpacing(spacing:float)	设置水平间距	None
setSpacing(spacing:float)	设置间距	None
verticalSpacing()	获取竖直间距	float
setVerticalSpacing(spacing:float)	设置竖直间距	None
itemAt(index:int)	根据索引获取图形控件	QGraphicsWidget
removeAt(index:int)	根据索引移除图形控件	None
count()	获取图形控件的数量	int

在表 5-28 中，Qt. AnchorPoint 的枚举值为 Qt. AnchorLeft、Qt. AnchorHorizontalCenter、Qt. AnchorRight、Qt. AnchorTop、Qt. AnchorVerticalCenter、Qt. AnchorBottom。

【实例 5-6】 创建一个窗口，该窗口包含 1 个图形控件。向该图形控件中添加 3 个控件，分别为一个显示图像的标签控件、两个按钮控件，使用线性布局垂直排列，代码如下：

```
# === 第 5 章 代码 demo6.py === #
import sys
from PySide6.QtWidgets import (QApplication,QWidget,QVBoxLayout,
   QGraphicsProxyWidget, QGraphicsScene,QGraphicsView,QPushButton,
   QGraphicsWidget,QGraphicsLinearLayout,QLabel)
from PySide6.QtGui import QPixmap
from PySide6.QtCore import Qt

class Window(QWidget):
   def __init__(self,parent = None):
      super().__init__(parent)
      self.setGeometry(200,200,580,280)
      self.setWindowTitle("图形控件的布局")
      # 创建图像视图控件
      view = QGraphicsView()
      # 创建图像场景控件
      scene = QGraphicsScene()
```

```
        #图像视图设置场景
        view.setScene(scene)
        #设置窗口布局
        vbox = QVBoxLayout(self)
        vbox.addWidget(view)
        #创建图形控件
        widget = QGraphicsWidget()
        widget.setFlags(QGraphicsWidget.ItemIsMovable|QGraphicsWidget.ItemIsSelectable)
        #向图像场景中添加图形控件
        scene.addItem(widget)
        #设置线性布局
        linear = QGraphicsLinearLayout(Qt.Vertical,widget)
        #创建标签控件并显示图像
        label1 = QLabel("标签控件")
        pix = QPixmap("D:\\Chapter5\\images\\cat1.png")
        pix = pix.scaled(380,220)              #缩放图像文件
        label1.setPixmap(pix)
        #创建两个按钮控件
        button1 = QPushButton("按钮控件 1")
        button2 = QPushButton("按钮控件 2")
        #创建代理控件,设置控件
        p1 = QGraphicsProxyWidget(); p1.setWidget(label1)
        p2 = QGraphicsProxyWidget(); p2.setWidget(button1)
        p3 = QGraphicsProxyWidget(); p3.setWidget(button2)
        #向线性布局中添加控件
        linear.addItem(p1);linear.addItem(p2);linear.addItem(p3)
        linear.setSpacing(5)
        linear.setStretchFactor(p1,1)
        linear.setStretchFactor(p2,2)
        linear.setStretchFactor(p3,2)


if __name__ == '__main__':
    app = QApplication(sys.argv)
    win = Window()
    win.show()
    sys.exit(app.exec())
```

运行结果如图 5-13 所示。

图 5-13　代码 demo6.py 的运行结果

【实例 5-7】 创建一个窗口，该窗口包含 1 个图形控件。向该图形控件中添加 9 个按钮控件，使用栅格布局排列，代码如下：

```
# === 第 5 章 代码 demo7.py === #
import sys
from PySide6.QtWidgets import (QApplication,QWidget,QVBoxLayout,
   QGraphicsProxyWidget, QGraphicsScene,QGraphicsView,QPushButton,
   QGraphicsWidget,QGraphicsGridLayout)
from PySide6.QtCore import Qt

class Window(QWidget):
   def __init__(self,parent = None):
      super().__init__(parent)
      self.setGeometry(200,200,580,280)
      self.setWindowTitle("图形控件的布局")
      #创建图像视图控件
      view = QGraphicsView()
      #创建图像场景控件
      scene = QGraphicsScene()
      #图像视图设置场景
      view.setScene(scene)
      #设置窗口布局
      vbox = QVBoxLayout(self)
      vbox.addWidget(view)
      #创建图形控件
      widget = QGraphicsWidget()
      widget.setFlags(QGraphicsWidget.ItemIsMovable|QGraphicsWidget.ItemIsSelectable)
      #向图像场景中添加图形控件
      scene.addItem(widget)
      #设置栅格布局
      grid = QGraphicsGridLayout(widget)
      #创建 9 个按钮控件
      button1 = QPushButton("按钮控件 1")
      button2 = QPushButton("按钮控件 2")
      button3 = QPushButton("按钮控件 3")
      button4 = QPushButton("按钮控件 4")
      button5 = QPushButton("按钮控件 5")
      button6 = QPushButton("按钮控件 6")
      button7 = QPushButton("按钮控件 7")
      button8 = QPushButton("按钮控件 8")
      button9 = QPushButton("按钮控件 9")
      #创建代理控件,设置控件
      p1 = QGraphicsProxyWidget(); p1.setWidget(button1)
      p2 = QGraphicsProxyWidget(); p2.setWidget(button2)
      p3 = QGraphicsProxyWidget(); p3.setWidget(button3)
      p4 = QGraphicsProxyWidget(); p4.setWidget(button4)
      p5 = QGraphicsProxyWidget(); p5.setWidget(button5)
      p6 = QGraphicsProxyWidget(); p6.setWidget(button6)
      p7 = QGraphicsProxyWidget(); p7.setWidget(button7)
      p8 = QGraphicsProxyWidget(); p8.setWidget(button8)
      p9 = QGraphicsProxyWidget(); p9.setWidget(button9)
```

```
        #向线性布局中添加控件
        grid.addItem(p1,0,0);grid.addItem(p2,0,1);grid.addItem(p3,0,2)
        grid.addItem(p4,1,0);grid.addItem(p5,1,1);grid.addItem(p6,1,2)
        grid.addItem(p7,2,0);grid.addItem(p8,2,1);grid.addItem(p9,2,2)
        grid.setSpacing(10)


if __name__ == '__main__':
    app = QApplication(sys.argv)
    win = Window()
    win.show()
    sys.exit(app.exec())
```

运行结果如图 5-14 所示。

图 5-14　代码 demo7.py 的运行结果


21min

5.3.4　图形效果类

在 PySide6 中,可以在图形项和图像视图控件的视口之间添加渲染通道,实现对图形项显示效果的特殊设置。图形效果类有 5 种,分别为 QGraphicsEffect(图形效果基类)、QGraphicsBlurEffect(模糊效果)、QGraphicsColorizeEffect(变色效果)、QGraphicsDropShadowEffect(阴影效果)、QGraphicsOpacityEffect(透明效果),这 5 个类的继承关系如图 5-15 所示。

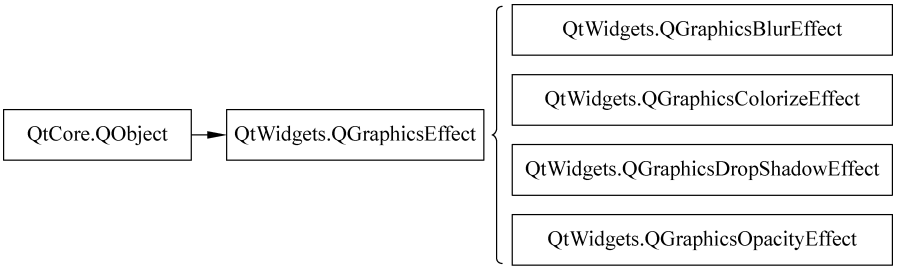

图 5-15　图形效果类的继承关系

1. 模糊效果类 QGraphicsBlurEffect

使用 QGraphicsBlurEffect 类可以创建模糊效果对象,使用模糊效果对象可以对图形项

的显示设置模糊效果。QGraphicsBlurEffect 类的构造函数如下：

```
QGraphicsBlurEffect(parent:QObject = None)
```

其中，parent 表示 QObject 类及其子类创建的实例对象。

QGraphicsBlurEffect 类的常用方法见表 5-29。

表 5-29 QGraphicsBlurEffect 类的常用方法

方法及参数类型	说 明	返回值的类型
[slot]setBlurHints(hints: QGraphicsBlurEffect. BlurHints)	设置模糊提示	None
[slot]setBlurRadius(blurRadius:float)	设置模糊半径，默认半径为 5 像素，模糊半径越大，图像越模糊	None
setEnabled(enable:bool)	设置是否激活图形效果	None
blurHints()	获取模糊提示	QGraphicsBlurEffect. BlurHints
blurRadius()	获取模糊半径	float

在表 5-29 中，QGraphicsBlurEffect. BlurHints 的枚举值为 QGraphicsBlurEffect. PerformanceHint(主要考虑渲染性能)、QGraphicsBlurEffect. QualityHint(主要考虑渲染质量)、QGraphicsBlurEffect. AnimationHint(用于渲染动画)。

QGraphicsBlurEffect 类的信号见表 5-30。

表 5-30 QGraphicsBlurEffect 类的信号

信号及参数类型	说 明
blurRadiusChanged(radius:float)	当模糊半径发生改变时发送信号
blurHintsChanged(hints:QGraphicsBlurEffect. BlurHints)	当模糊提示发生改变时发送信号

2. 变色效果类 QGraphicsColorizeEffect

使用 QGraphicsColorizeEffect 类可以创建变色效果对象，使用模糊效果对象可以对图形项的显示设置变色效果。QGraphicsColorizeEffect 类的构造函数如下：

```
QGraphicsColorizeEffect(parent:QObject = None)
```

其中，parent 表示 QObject 类及其子类创建的实例对象。

QGraphicsColorizeEffect 类的常用方法见表 5-31。

表 5-31 QGraphicsColorizeEffect 类的常用方法

方法及参数类型	说 明	返回值的类型
[slot]setColor(Union[QColor,Qt. GlobalColor,str])	设置着色用的颜色，默认颜色为浅蓝色 QColor(0,0,192)	None
[slot]setStrength(strength:float)	设置着色强度	None
color()	获取着色用的颜色	QColor
strength()	获取着色强度	float

QGraphicsColorizeEffect 类的信号见表 5-32。

表 5-32 QGraphicsColorizeEffect 类的信号

信号及参数类型	说 明
colorChanged(color:QColor)	当颜色发生改变时发送信号
strengthChanged(strength:float)	当强度发生改变时发送信号

3. 阴影效果类 QGraphicsDropShadowEffect

使用 QGraphicsDropShadowEffect 类可以创建阴影效果对象,使用阴影效果对象可以对图形项的显示设置阴影效果。QGraphicsDropShadowEffect 类的构造函数如下:

```
QGraphicsDropShadowEffect(parent:QObject = None)
```

其中,parent 表示 QObject 类及其子类创建的实例对象。

QGraphicsDropShadowEffect 类的常用方法见表 5-33。

表 5-33 QGraphicsDropShadowEffect 类的常用方法

方法及参数类型	说 明	返回值的类型
[slot]setBlurRadius(blurRadius:float)	设置模糊半径	None
[slot]setColor(Union[QColor,Qt.GlobalColor,str])	设置阴影颜色	None
[slot]setOffset(d:float)	设置阴影的 x 和 y 偏移量	None
[slot]setOffset(dx:float,dy:float)	设置阴影的 x 和 y 偏移量	None
[slot]setOffset(ofs:Union[QPoint,QPointF])	设置阴影的偏移量	None
[slot]setXOffset(dx:float)	设置阴影的 x 偏移量	None
[slot]setYOffset(dy:float)	设置阴影的 y 偏移量	None
blurRadius()	获取模糊半径	float
color()	获取阴影颜色	QColor
offset()	获取阴影的偏移量	QPointF
xOffset()	获取阴影的 x 偏移量	float
yOffset()	获取阴影的 y 偏移量	float

QGraphicsDropShadowEffect 类的信号见表 5-34。

表 5-34 QGraphicsDropShadowEffect 类的信号

信号及参数类型	说 明
blurRadiusChanged(blurRadius:float)	当模糊半径发生改变时发送信号
colorChanged(color:QColor)	当阴影颜色发生改变时发送信号
offsetChanged(offset:QPointF)	阴影偏移量发生改变时发送信号

4. 透明效果类 QGraphicsOpacityEffect

使用 QGraphicsOpacityEffect 类可以创建透明效果对象,使用透明效果对象可以对图形项的显示设置透明效果。QGraphicsOpacityEffect 类的构造函数如下:

```
QGraphicsOpacityEffect(parent:QObject = None)
```

其中，parent 表示 QObject 类及其子类创建的实例对象。

QGraphicsOpacityEffect 类的常用方法见表 5-35。

表 5-35　QGraphicsOpacityEffect 类的常用方法

方法及参数类型	说　明	返回值类型
[slot]setOpacity(opacity:float)	设置不透明度	None
[slot]setOpacityMask(Union[QBrush, Qt.BrushStyle, Qt.GlobalColor, QColor, QGradient, QImage, QPixmap])	设置遮掩画刷	None
opacity()	获取不透明度	float
opacityMask()	获取遮掩画刷	QBrush

QGraphicsOpacityEffect 类的信号见表 5-36。

表 5-36　QGraphicsOpacityEffect 类的信号

信号及参数类型	说　明
opacityChanged(opacity:float)	当不透明度发生改变时发送信号
opacityMaskChanged(mask:QBrush)	当遮掩画刷发生改变时发送信号

【实例 5-8】 创建一个窗口，该窗口包含 5 个按钮、1 个图像视图控件。5 个按钮的功能分别为打开图像、实现模糊效果、实现变色效果、实现阴影效果、实现透明效果，代码如下：

```
# === 第 5 章 代码 demo8.py === #
import sys,os
from PySide6.QtWidgets import (QApplication,QWidget,QVBoxLayout,
    QHBoxLayout,QGraphicsScene,QGraphicsView,QPushButton,
    QGraphicsBlurEffect,QGraphicsColorizeEffect,
    QGraphicsDropShadowEffect,QGraphicsOpacityEffect,QFileDialog,
    QGraphicsPixmapItem)
from PySide6.QtGui import QPixmap,QLinearGradient
from PySide6.QtCore import Qt

class Window(QWidget):
    def __init__(self,parent=None):
        super().__init__(parent)
        self.resize(580,280)
        self.setWindowTitle("图形效果")
        #用于保存图像文件
        self.pixmapItem=None
        self.view=QGraphicsView()          #图像视图控件
        self.scene=QGraphicsScene()        #图像场景
        self.view.setScene(self.scene)     #在图像视图中设置场景
        #创建多个按钮
        self.btnOpen=QPushButton("打开图像")
        self.btnBlur=QPushButton("模糊效果")
        self.btnColor=QPushButton("变色效果")
        self.btnShadow=QPushButton("阴影效果")
        self.btnOpacity=QPushButton("透明效果")
        #设置布局
        hbox=QHBoxLayout()
```

```
        hbox.addWidget(self.btnOpen)
        hbox.addWidget(self.btnBlur)
        hbox.addWidget(self.btnColor)
        hbox.addWidget(self.btnShadow)
        hbox.addWidget(self.btnOpacity)
        vbox = QVBoxLayout(self)
        vbox.addLayout(hbox)
        vbox.addWidget(self.view)
        #使用信号/槽
        self.btnOpen.clicked.connect(self.btn_open)
        self.btnBlur.clicked.connect(self.btn_blur)
        self.btnColor.clicked.connect(self.btn_color)
        self.btnShadow.clicked.connect(self.btn_shadow)
        self.btnOpacity.clicked.connect(self.btn_opacity)
        #设置按钮处于失效状态
        self.btnBlur.setEnabled(False)
        self.btnColor.setEnabled(False)
        self.btnShadow.setEnabled(False)
        self.btnOpacity.setEnabled(False)

    def btn_open(self):
        (fileName,filter) = QFileDialog.getOpenFileName(self,caption = "打开图像文件",filter =
"图像(*.png *.bmp *.jpg *.jpeg)")
        if os.path.exists(fileName):
            if self.pixmapItem != None:
                self.scene.removeItem(self.pixmapItem)
            pix = QPixmap(fileName)
            self.pixmapItem = QGraphicsPixmapItem(pix)
            self.scene.addItem(self.pixmapItem)
            self.btnBlur.setEnabled(True)
            self.btnColor.setEnabled(True)
            self.btnShadow.setEnabled(True)
            self.btnOpacity.setEnabled(True)
        else:
            if self.pixmapItem == None:
                self.btnBlur.setEnabled(False)
                self.btnColor.setEnabled(False)
                self.btnShadow.setEnabled(False)
                self.btnOpacity.setEnabled(False)

    def btn_blur(self):
        self.effect = QGraphicsBlurEffect()
        self.effect.setBlurRadius(10)
        self.effect.setBlurHints(QGraphicsBlurEffect.QualityHint)
        self.pixmapItem.setGraphicsEffect(self.effect)

    def btn_color(self):
        self.effect = QGraphicsColorizeEffect()
        self.effect.setColor(Qt.blue)
        self.effect.setStrength(10)
        self.pixmapItem.setGraphicsEffect(self.effect)
```

```
    def btn_shadow(self):
        self.effect = QGraphicsDropShadowEffect()
        self.pixmapItem.setGraphicsEffect(self.effect)

    def btn_opacity(self):
        rect = self.pixmapItem.boundingRect()
        linear = QLinearGradient(rect.topLeft(),rect.bottomLeft())
        linear.setColorAt(0.11,Qt.transparent)
        linear.setColorAt(0.49,Qt.black)
        linear.setColorAt(0.88,Qt.white)
        self.effect = QGraphicsOpacityEffect()
        self.effect.setOpacityMask(linear)
        self.pixmapItem.setGraphicsEffect(self.effect)


if __name__ == '__main__':
    app = QApplication(sys.argv)
    win = Window()
    win.show()
    sys.exit(app.exec())
```

运行结果如图 5-16 所示。

图 5-16　代码 demo8.py 的运行结果

5.4　小结

本章首先介绍了 Graphics/View 绘图框架，也就是使用图像视图类、图像场景类、图形项类绘制图像，而且图像视图、图像场景、图形项都有各自的坐标系；其次介绍了图像视图类、图像场景类、图形项类的构造函数、常用方法、信号；最后介绍了向图像场景中添加控件、设置图形效果的方法。使用 Graphics/View 框架，不仅可以绘制图像，还可以向其中添加控件，而且可将这些控件设置为可移动控件。