第7章 Qt 5图形视图框架

7.1 图形视图体系结构

7.1.1 Graphics View的特点

（1）Graphics View框架结构中，系统可以利用Qt绘图系统的反锯齿、OpenGL工具来改善绘图性能。

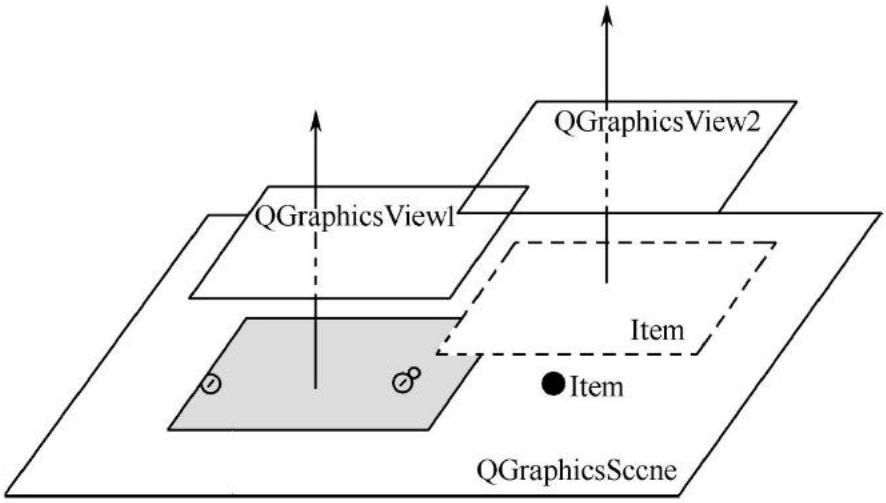
（2）Graphics View支持事件传播体系结构，可以使图元在场景（scene）中的交互能力提高一倍，图元能够处理键盘事件和鼠标事件。其中，鼠标事件包括鼠标按下、移动、释放和双击，还可以跟踪鼠标的移动。

（3）在Graphics View框架中，通过二元空间划分树（Binary Space Partitioning，BSP）提供快速的图元查找，这样就能够实时地显示包含上百万个图元的大场景。

7.1.2 Graphics View的三元素

Graphics View框架结构主要包含三个类，场景类（QGraphicsScene）、视图类（QGraphicsView）和图元类（QGraphicsItem），统称为“三元素”。

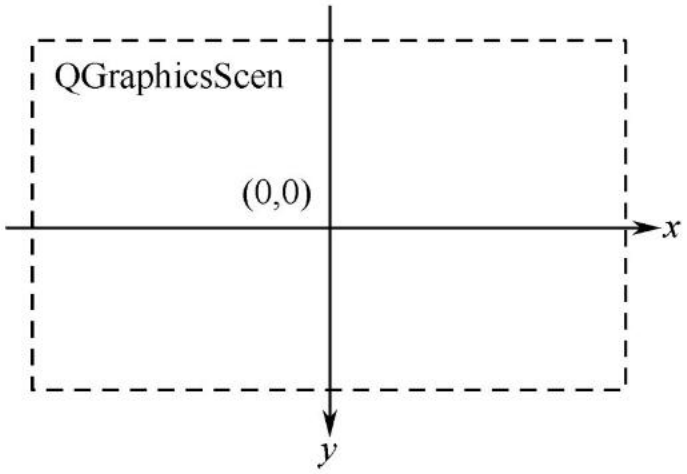
它们三者之间的关系如图7.1所示。



7.1.3 GraphicsView的坐标系统

1. 场景坐标

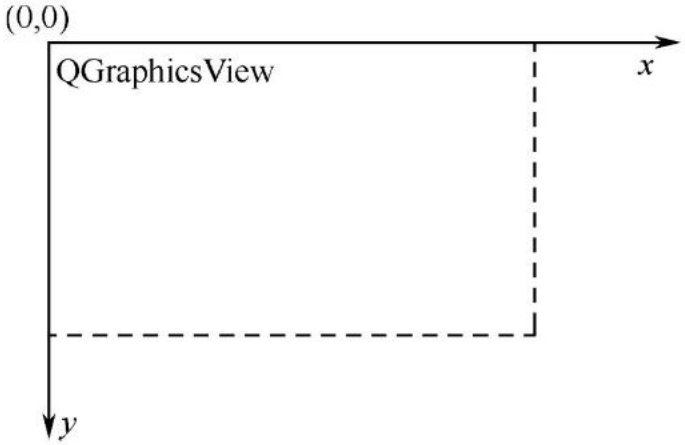
QGraphicsScene类的坐标系以中心为原点（0,0），如图7.2所示。



7.1.3 GraphicsView的坐标系统

2. 视图坐标

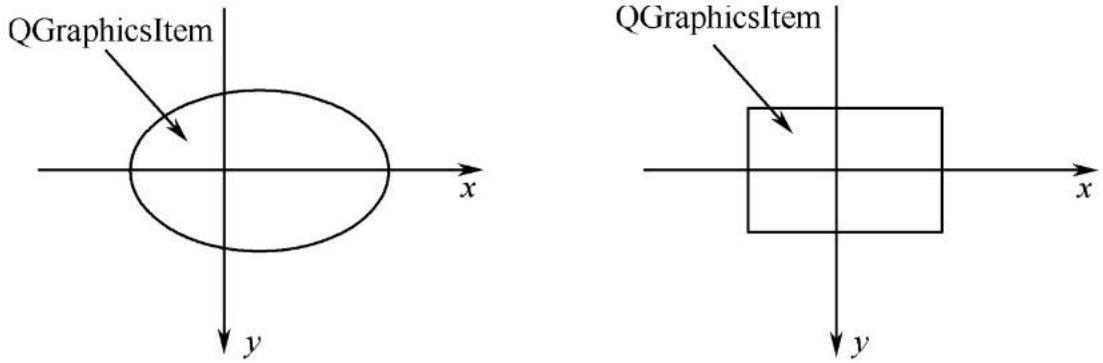
QGraphicsView类继承自QWidget类，因此它与其他的QWidget类一样，以窗口的左上角作为自己坐标系的原点，如图7.3所示。



7.1.3 GraphicsView的坐标系统

3. 图元坐标

QGraphicsItem类的坐标系，若在调用QGraphicsItem类的paint()函数重绘图元时，则以此坐标系为基准，如图7.4所示。



（a）　　（b）

7.1.3 GraphicsView的坐标系统

Graphics View框架提供了多种坐标变换函数，见表7.1。

映 射 函 数	转 换 类 型
QGraphicsView::mapToScene()	视图到场景
QGraphicsView::mapFromScene()	场景到视图
QGraphicsItem:: mapFromScene()	场景到图元
QGraphicsItem:: mapToScene()	图元到场景
QGraphicsItem:: mapToParent()	子图元到父图元
QGraphicsItem:: mapFromParent()	父图元到子图元
QGraphicsItem:: mapToItem()	本图元到其他图元
QGraphicsItem:: mapFromItem()	其他图元到本图元

7.2 图形视图示例

7.2.1 飞舞的蝴蝶例子

（1）新建Qt Gui应用，项目名为“Butterfly”，基类选择“QMainWindow”，类名命名默认为“MainWindow”，取消“创建界面”复选框的选中状态。单击“下一步”按钮，最后单击“完成”按钮，完成该项目工程的建立。

（2）在“Butterfly”项目名上单击鼠标右键，在弹出的快捷菜单中选择“添加新文件...”菜单项，在弹出的对话框中选择“C++ 类”选项。单击“选择”按钮，弹出“C++ 类向导”对话框，在“基类”后面的下拉列表框中选择基类名“QObject”，在“类名”后面的文本框中输入类的名称“Butterfly”。

（3）单击“下一步”按钮，单击“完成”按钮，添加文件“butterfly.h”和“butterfly. cpp”。

（4）Butterfly类继承自QObject类、QGraphicsItem类，在头文件“butterfly.h”中完成的代码具体内容。

7.2.1 飞舞的蝴蝶例子

（5）在源文件“butterfly. cpp”中完成的代码具体内容如下：

```
#include "butterfly.h"
#include <math.h>

const static double PI=3.1416;

Butterfly::Butterfly(QObject *parent)
{
    up = true;
    pix_up.load("up.png");
    pix_down.load("down.png");

    startTimer(100);
}
```

7.2.1 飞舞的蝴蝶例子

boundingRect()函数为图元限定区域范围。此范围是以图元自身的坐标系为基础设定的。具体实现代码内容如下：

```
QRectF Butterfly::boundingRect() const
{
    qreal adjust =2;
    return QRectF(-pix_up.width()/2-adjust,-pix_up.height()/2-adjust,
                  pix_up.width()+adjust*2,pix_up.height()+adjust*2);
}
```

7.2.1 飞舞的蝴蝶例子

在重画函数paint()中，首先判断当前已显示的图片是pix_up还是pix_down。实现蝴蝶翅膀上下飞舞效果时，若当前显示的是pix_up图片，则重绘pix_down图片，反之亦然。具体实现代码内容如下：

```
void Butterfly::paint(QPainter *painter, const QStyleOptionGraphicsItem
*option, QWidget *widget)
{
    if(up)
    {
        painter->drawPixmap(boundingRect().topLeft(),pix_up);
        up=!up;
    }
    else
    {
        painter->drawPixmap(boundingRect().topLeft(),pix_down);
        up=!up;
    }
}
```

7.2.1 飞舞的蝴蝶例子

定时器的timerEvent()函数实现蝴蝶的飞舞，具体实现代码内容如下：

```
void Butterfly::timerEvent(QTimerEvent *)
{
    //边界控制
    qreal edgex=scene()->sceneRect().right()+boundingRect().width()/2;
    qreal edgetop=scene()->sceneRect().top()+boundingRect().height()/2;
    qreal edgebottom=scene()->sceneRect().bottom()+boundingRect(). height()/2;

    if(pos().x()>=edgex)
        setPos(scene()->sceneRect().left(),pos().y());
    if(pos().y()<=edgetop)
        setPos(pos().x(),scene()->sceneRect().bottom());
    if(pos().y()>=edgebottom)
        setPos(pos().x(),scene()->sceneRect().top());

    angle+=(qrand()%10)/20.0;
    qreal dx=fabs(sin(angle*PI)*10.0);
    qreal dy=(qrand()%20)-10.0;

    setPos(mapToParent(dx,dy));
}
```

7.2.1 飞舞的蝴蝶例子

（6）完成了蝴蝶图元的实现后，在源文件“main.cpp”中将它加载到场景中，并关联一个视图，具体实现代码内容如下：

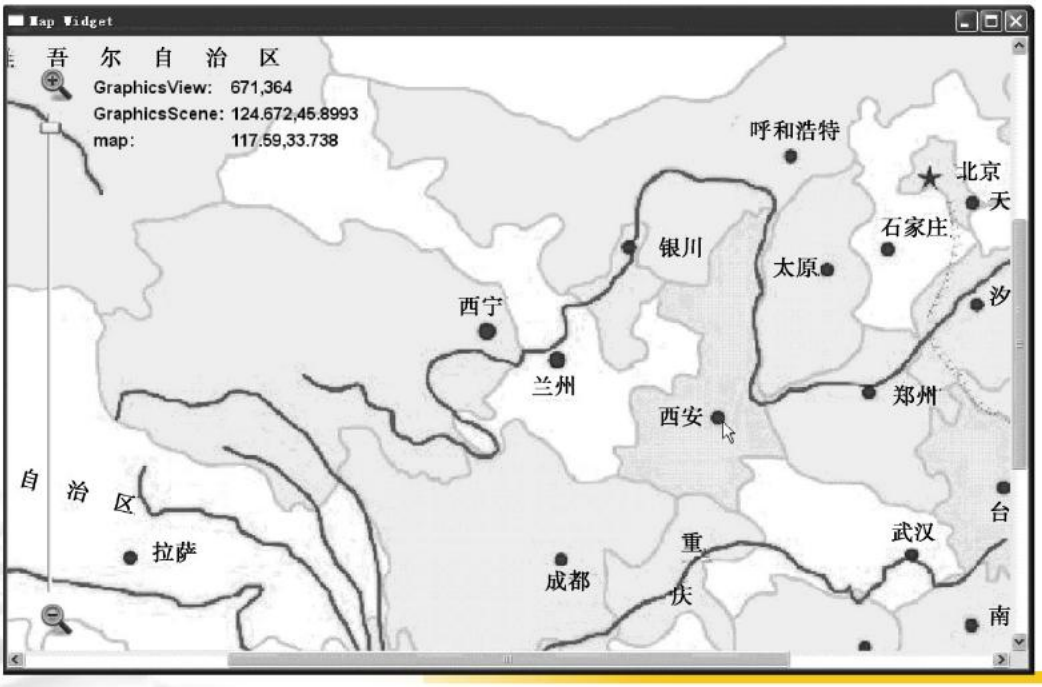
```
#include <QApplication>
#include "butterfly.h"
#include <QGraphicsScene>

int main(int argc,char* argv[])
{
    QApplication a(argc,argv);

    QGraphicsScene *scene = new QGraphicsScene;
    scene->setSceneRect(QRectF(-200,-200,400,400));

    Butterfly *butterfly = new Butterfly;
    butterfly->setPos(-100,0);

    scene->addItem(butterfly);

    QGraphicsView *view = new QGraphicsView;
    view->setScene(scene);
    view->resize(400,400);
    view->show();
    return a.exec();
}
```

7.2.1 飞舞的蝴蝶例子

（7）运行程序，将程序中用到的图片保存到该工程的D:\Qt\CH7\CH701\build-Butterfly- Desktop_Qt_5_0_2_MinGW_32bit-Debug文件夹中，运行结果如图7.5所示。

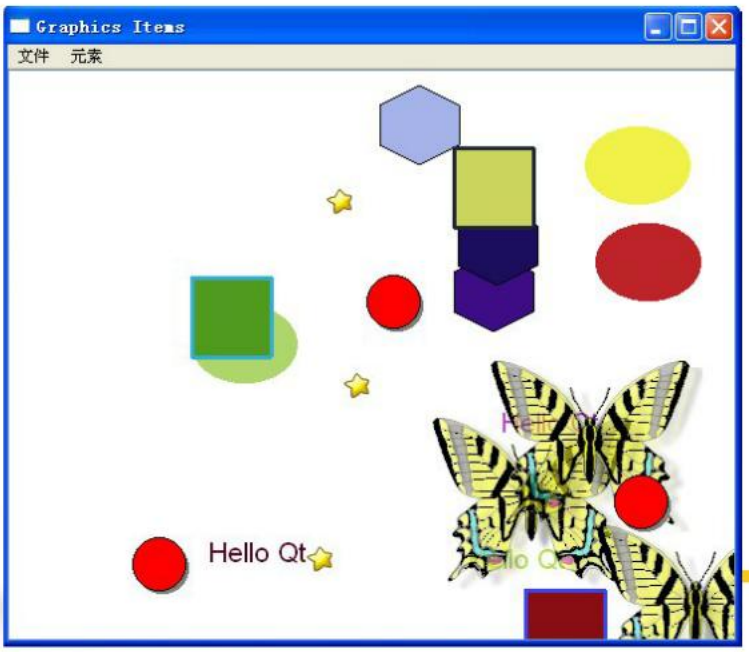
7.2.2 地图浏览器例子

通过实现一个地图浏览器的基本功能（包括地图的浏览、放大、缩小，以及显示各点的坐标等）的例子，如图7.6所示。



7.2.3 各种图元（GraphicsItem）的创建

通过实现一个在其中显示各种类型QGraphicsItem的窗体例子（如图7.7所示）。介绍如何使用Qt预定义的各种标准的QGraphicsItem类型（如QGraphicsEllipseItem、QGraphicsRectItem等），以及自定义的QGraphicsItem类型（如本例中不停闪烁的圆及来回移动的星星等）来创建图元（详细内容见代码CH703）。



7.2.4 图元的旋转、缩放、切变和位移

通过此实例介绍如何实现图元（QGraphicsItem）的旋转、缩放、切变及位移等各种变形操作，如图7.10所示（详细内容见代码CH704）。