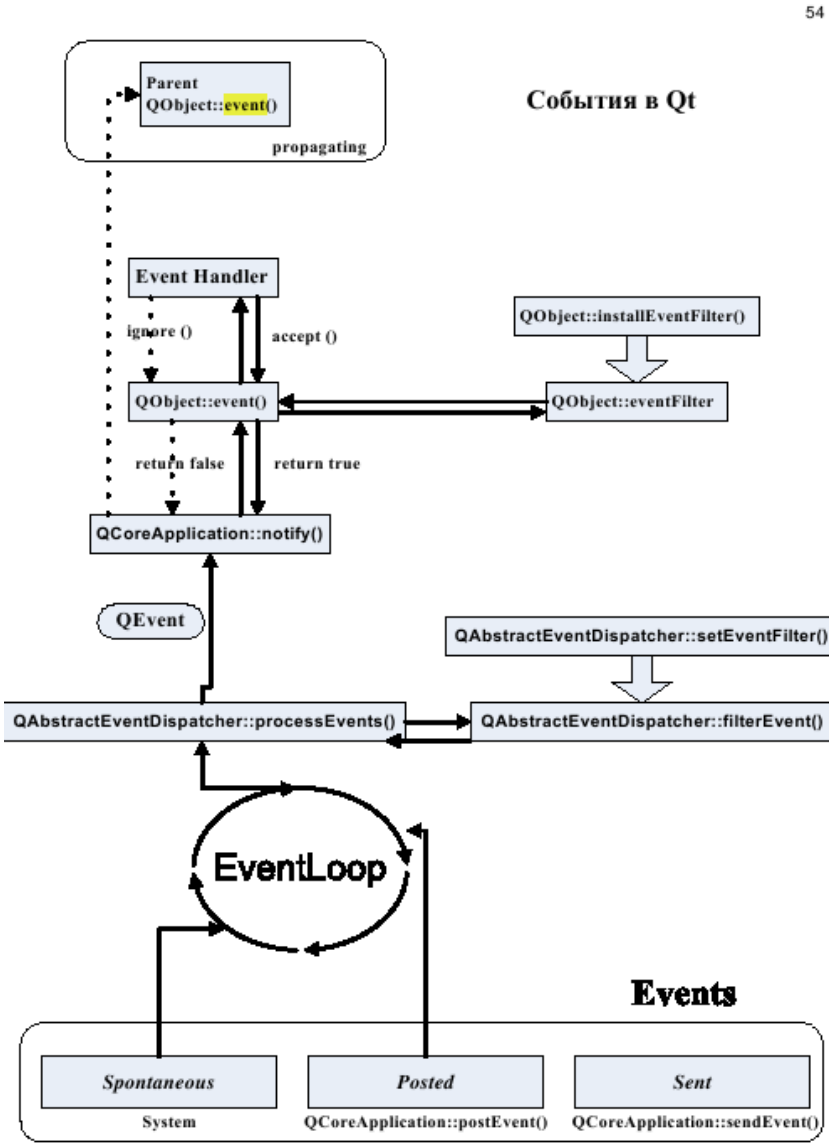
Технология ООП

Санкт-Петербургский государственный политехнический университет

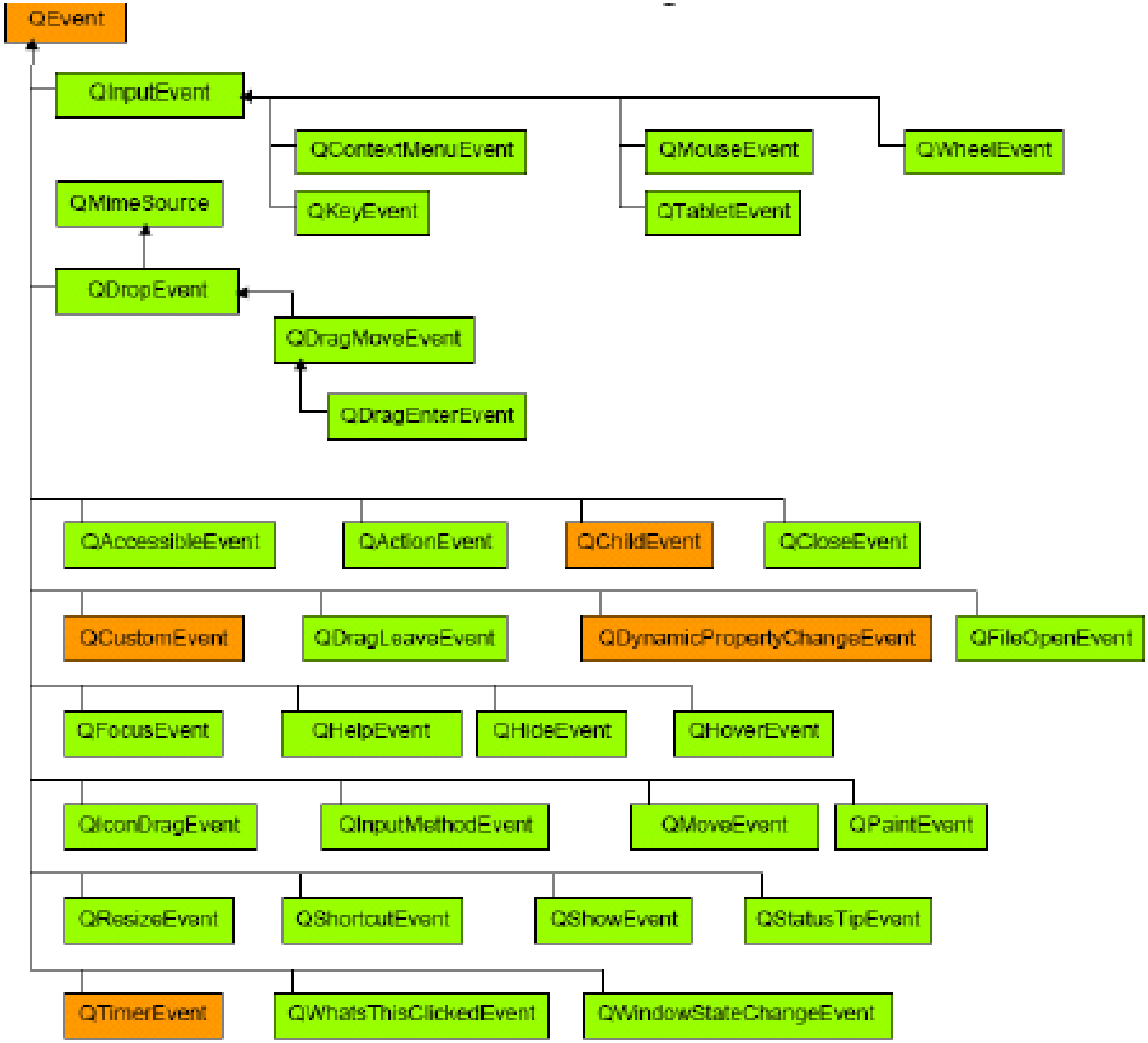
22 ноября 2011

События в Qt

В Qt событие — это экземпляр класса QEvent. В Qt редко приходится использовать события напрямую, т.к. большинство виджетов сами генерируют сигналы в ответ на любое существенное событие.


54
Parent
QObject::event()
propagating
События в Qt
Event Handler
ignore ()
accept ()
QObject::installEventFilter()
QObject::event()
QObject::eventFilter
return false
return true
QCoreApplication::notify()
QEvent
QAbstractEventDispatcher::setEventFilter()
QAbstractEventDispatcher::processEvents()
QAbstractEventDispatcher::filterEvent()
EventLoop
Events
Spontaneous
Posted
Sent
System
QCoreApplication::postEvent()
QCoreApplication::sendEvent()

События в Qt


QEvent
QInputEvent
QContextMenuEvent
QMouseEvent
QWheelEvent
QMimeSource
QKeyEvent
QTabletEvent
QDropEvent
QDragMoveEvent
QDragEnterEvent
QAccessibleEvent
QActionEvent
QChildEvent
QCloseEvent
QCustomEvent
QDragLeaveEvent
QDynamicPropertyChangeEvent
QFileOpenEvent
QFocusEvent
QHelpEvent
QHideEvent
QHoverEvent
QIconDragEvent
QInputMethodEvent
QMoveEvent
QPaintEvent
QResizeEvent
QShortcutEvent
QShowEvent
QStatusTipEvent
QTimerEvent
QWhatsThisClickedEvent
QWindowStateChangeEvent

События в Qt

События клавиатуры обрабатываются путем переопределения функций keyPressEvent() и keyReleaseEvent().

```
void MyClass::keyPressEvent(QKeyEvent *event)
{
switch(event->key())
{
case Qt::Key_Space:
qDebug("Space pressed");
break;
...
default:
QWidget::keyPressEvent(event);
}
}
```

QEvent

```
void accept ()
void ignore ()
bool isAccepted () const
void setAccepted ( bool accepted )
bool spontaneous () const
Type type () const
int registerEventType ( int hint = -1 ) [static]
```

QCoreApplication

```
void postEvent ( QObject * receiver, QEvent * event
    )
void processEvents ( QEventLoop::ProcessEventsFlags
    flags = QEventLoop::AllEvents )
void removePostedEvents ( QObject * receiver )
bool sendEvent ( QObject * receiver, QEvent * event
    )
void sendPostedEvents ( QObject * receiver, int
   event_type )
void sendPostedEvents ()
```

Обработчики событий

QObject:

```
virtual bool event ( QEvent * e )
virtual bool eventFilter ( QObject * watched,
    QEvent * event )
void installEventFilter ( QObject * filterObj )
```

Обработчики событий

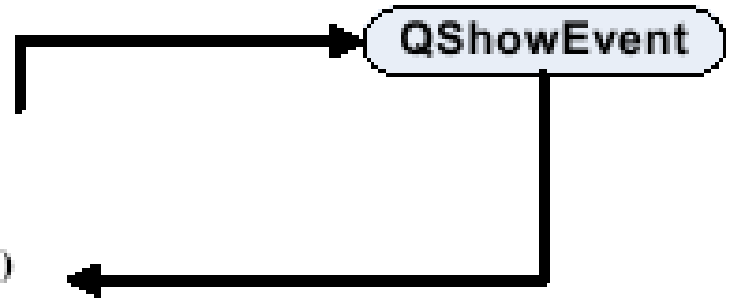
```
bool QObject::event(QEvent *event)
{
    switch (e->type()) {
    case QEvent::Timer:
        timerEvent((QTimerEvent*)e);
        break;
    case QEvent::ChildAdded:
    case QEvent::ChildInserted:
    case QEvent::ChildRemoved:
        childEvent((QChildEvent*)e);
        break;
    case QEvent::DeferredDelete:
        delete this;
        break;
    }
    return true;
 }
```

Обработчики событий

```
bool QWidget::event(QEvent *event)
  {
    switch (e->type()) {
    case QEvent::KeyPress:
        keyPressEvent((QKeyEvent *)event);
        if (!((QKeyEvent *)event)->isAccepted())
            return false;
        break;
    case QEvent::KeyRelease:
        keyReleaseEvent((QKeyEvent *)event);
        if (!((QKeyEvent *)event)->isAccepted())
            return false;
        break;
        ...
    }
    return true;
  }
```

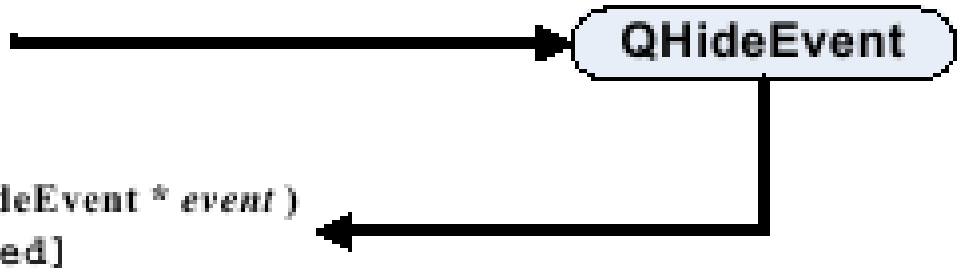
События в Qt

- virtual void setVisible (bool *visible*)
- void show () [slot]



- void showEvent (QShowEvent * *event*)
  [virtual protected]

- void hide () [slot]



- void :hideEvent (QHideEvent * *event*)
  [virtual protected]

События в Qt

- **bool close () [slot]**
- **void closeEvent (QCloseEvent * *event*) [virtual protected]**

Установка фильтров событий

```
void MyLineEdit::keyPressEvent(QKeyEvent *event)
{
if(event->key() == Qt::Key_Space) { focusNextChild
    (); }
else { QLineEdit::keyPressEvent(event); }
}
```

Установка фильтров событий

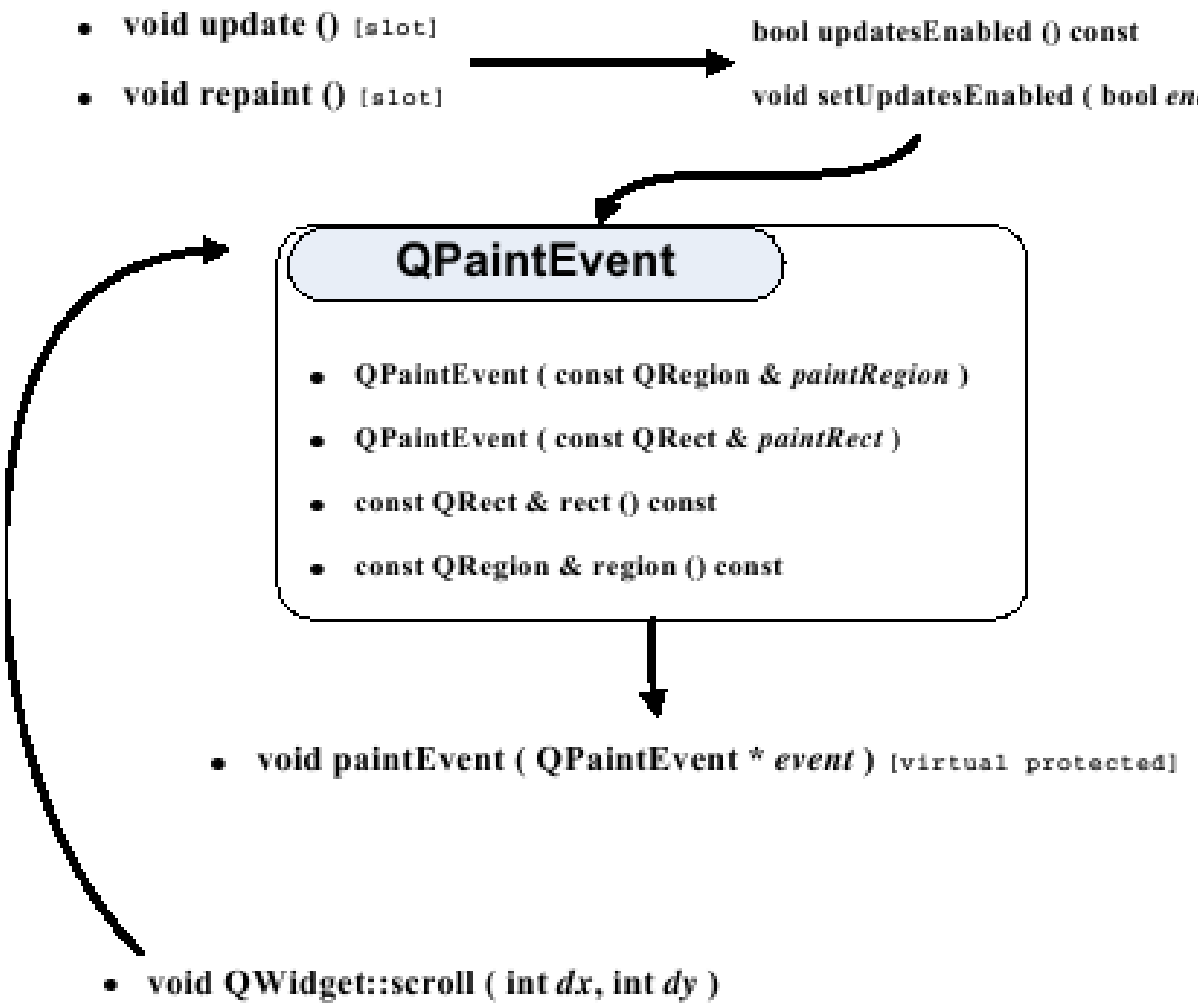
Настройка фильтров событий состоит из следующих этапов:

- Регистрация объекта-перехватчика с целевым объектом посредством вызова функции installEventFilter().
- Обработка события в функции eventFilter() перехватчика

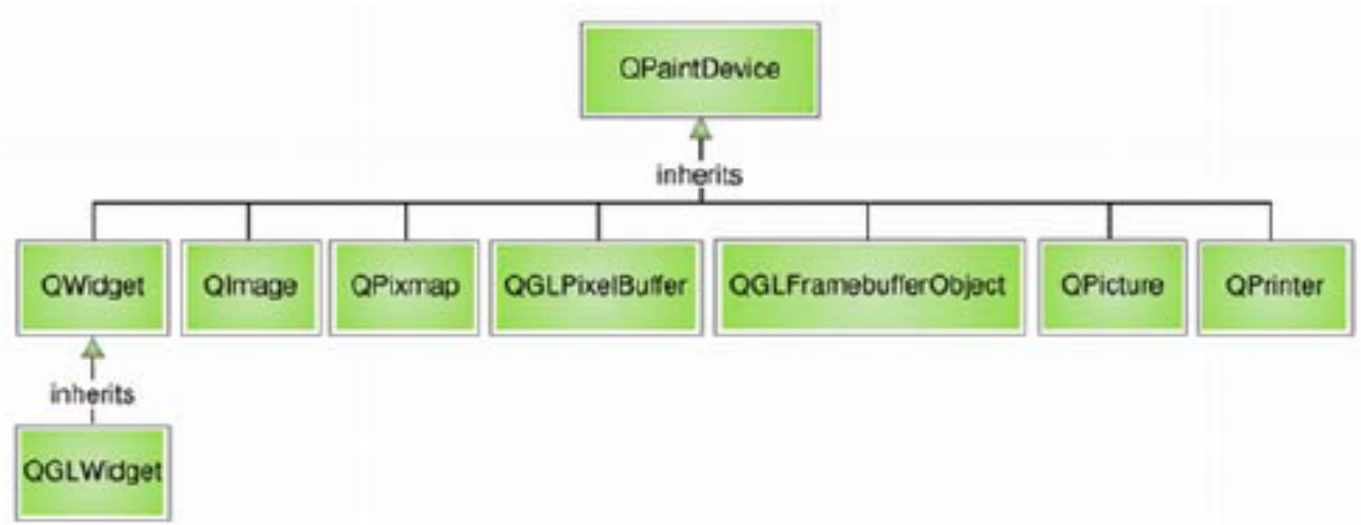
Установка фильтров событий

```
MyClass::MyClass(QWidget *parent)
{
firstNameEdit->installEventFilter(this);
secondNameEdit->installEventFilter(this);
cityEdit->installEventFilter(this);
}
bool MyClass::eventFilter(QObject *target, QEvent *
    event)
{
if(target == firstNameEdit || target ==
    secondNameEdit || ... )
{
QKeyEvent *keyEvent = static_cast<QKeyEvent*>(event
    );
if(keyEvent->key() == Qt::Key_Space) {
    focusNextChild(); return true; }
}
return QWidget::eventFilter(target, event);
}
```

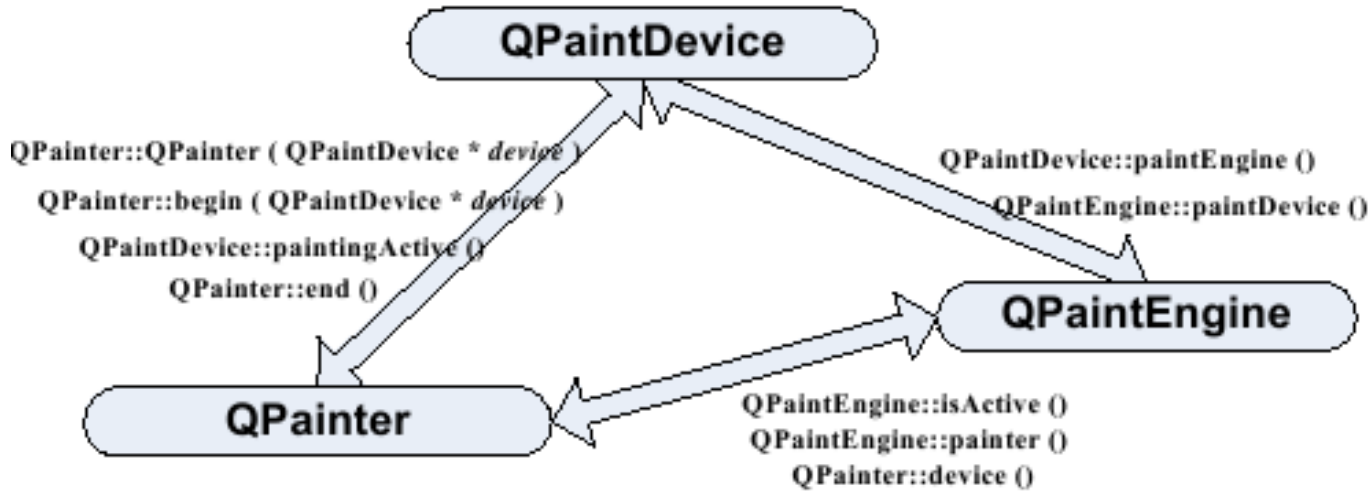
События в Qt


void update () [slot]
void repaint () [slot]
bool updatesEnabled () const
void setUpdatesEnabled (bool enable)
QPaintEvent
QPaintEvent (const QRegion & paintRegion)
QPaintEvent (const QRect & paintRect)
const QRect & rect () const
const QRegion & region () const
void paintEvent (QPaintEvent * event) [virtual protected]
void QWidget::scroll (int dx, int dy)

Вывод графики



Вывод графики



Вывод графики

QPainter может использоваться для вычерчивания на различных устройствах, таких как QWidget, QPixmap, QImage, QPrinter и т.п., что позволяет использовать один программный код для отображения на экран, получения изображения и печати отчетов.

QPaintEngine

```
enum QPaintEngine::Type
    QPaintEngine::X11
    QPaintEngine::Windows
    QPaintEngine::MacPrinter
    QPaintEngine::CoreGraphics
    QPaintEngine::QuickDraw
    QPaintEngine::QWindowSystem
    QPaintEngine::PostScript
    QPaintEngine::OpenGL
    QPaintEngine::Picture
    QPaintEngine::SVG
    QPaintEngine::Raster
    QPaintEngine::Direct3D
    QPaintEngine::Pdf
    QPaintEngine::User
```

QPaintEngine

```
virtual void drawEllipse ( const QRect & rect )
virtual void drawLines ( const QLine * lines, int
    lineCount )
virtual void drawPath ( const QPainterPath & path )
virtual void drawImage ( const QRectF & rectangle,
    const QImage & image, const QRectF & sr, Qt::
    ImageConversionFlags flags = Qt::AutoColor )
virtual void drawPoints ( const QPoint * points,
    int pointCount )
virtual void drawPolygon ( const QPoint * points,
    int pointCount, PolygonDrawMode mode )
```

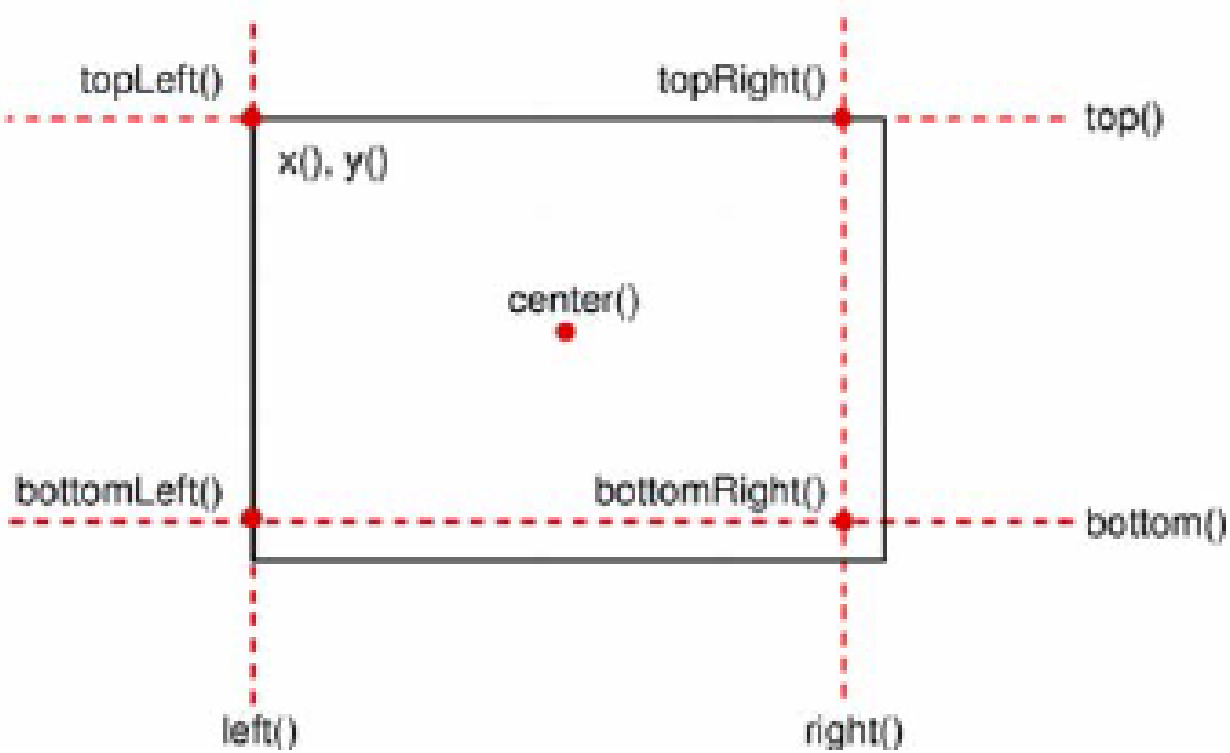
QPoint(F)

```
int manhattanLength () const
int (qreal) & rx ()
(int | qreal) & ry ()
void setX ((int | qreal)x )
void setY ( int y )
int x () const
int y () const
```

QLine(F)

```
void setP1 ( const QPoint & p1 )
void setP2 ( const QPoint & p2 )
void setLine ( int x1, int y1, int x2, int y2 )
void setPoints ( const QPoint & p1, const QPoint &
    p2)
void translate ( const QPoint & offset )
QLine translated ( const QPoint & offset ) const
qreal angle () const (F)
qreal angleTo ( const QLineF & line ) const (F)
qreal length () const (F)
QLine toLine () const (F)
```

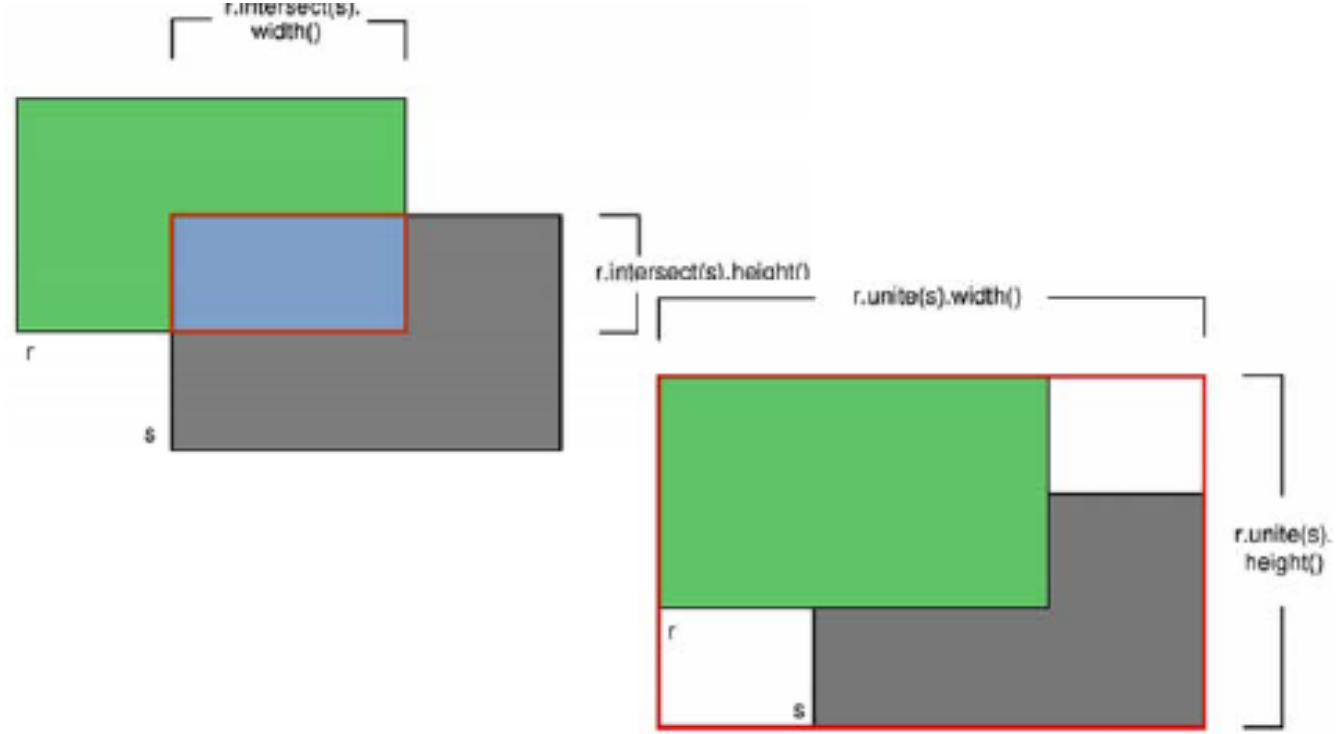
QRect(F)


topLeft()
topRight()
top()
x(), y()
center()
bottomLeft()
bottomRight()
bottom()
left()
right()

QRect(F)

```
void adjust ( int dx1, int dy1, int dx2, int dy2 )
QRect adjusted ( int dx1, int dy1, int dx2, int dy2
    ) const
bool contains ( const QPoint & point, bool proper =
    false ) const
QRect intersected ( const QRect & rectangle ) const
bool intersects ( const QRect & rectangle ) const
QRect united ( const QRect & rectangle ) const
```

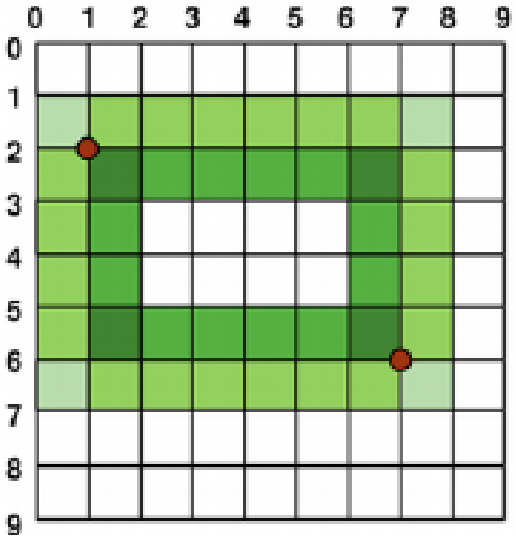
QRect(F)


r.intersect(s).
width()
r.intersect(s).height()
r
s
r.unite(s).width()
r.unite(s).
height()
r
s

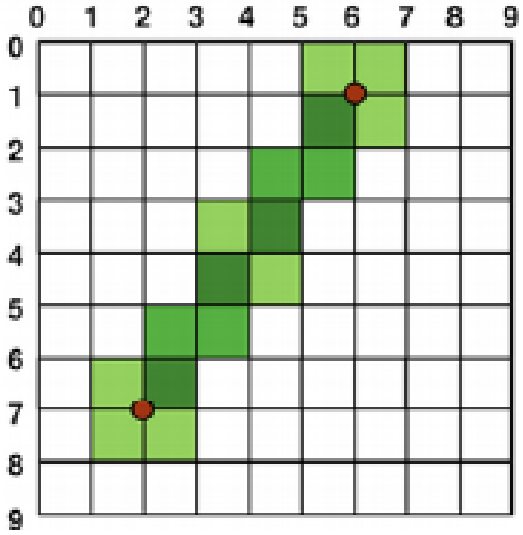
QPolygon(F)

```
QRect united ( const QRect & rectangle ) const
QRectF boundingRect ()const
bool containsPoint ( const QPointF & point, Qt::
    FillRule fillRule ) const
QPolygonF intersected ( const QPolygonF & r ) const
bool isClosed () const
QPolygonF subtracted ( const QPolygonF & r ) const
QPolygon toPolygon () const (F)
void translate ( const QPointF & offset )
void translate ( qreal dx, qreal dy )
QPolygonF united ( const QPolygonF & r ) const
```

Рендеринг




```
QPainter painter(this);
painter.setRenderHint( QPainter::Antialiasing);
painter.setPen(Qt::darkGreen);
painter.drawRect(1, 2, 6, 4);
```



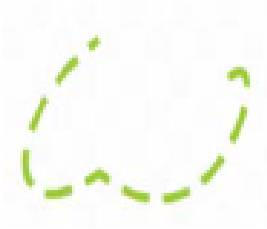

```
QPainter painter(this);
 painter.setRenderHint(QPainter::Antialiasing);
painter.setPen(Qt::darkGreen);
painter.drawLine(2, 7, 6, 1);
```

QPainter::RenderHint

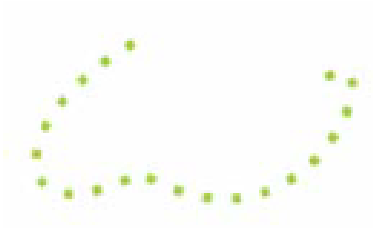
```
enum QPainter::RenderHint
flags QPainter::RenderHints
    QPainter::Antialiasing
    QPainter::TextAntialiasing
    QPainter::SmoothPixmapTransform
    QPainter::HighQualityAntialiasing

RenderHints renderHints () const
void setRenderHint ( RenderHint hint, bool on =
    true )
void setRenderHints ( RenderHints hints, bool on =
    true )
```

QPen

Qt::SolidLine



Qt::DashLine



Qt::DotLine

Qt::DashDotLine

Qt::DashDotDotLine

Qt::CustomDashLine

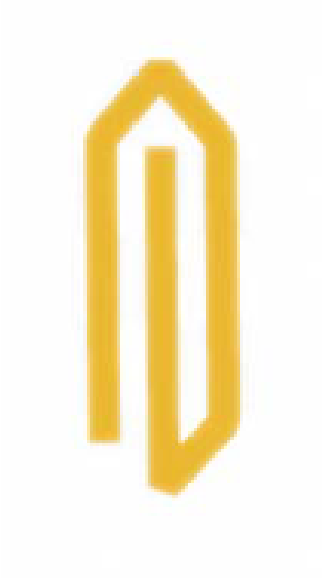
QPen

Qt::SquareCap

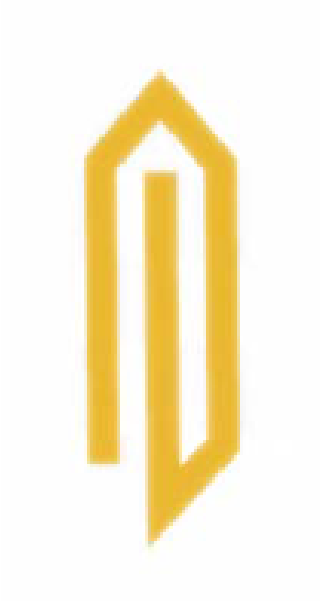
Qt::FlatCap

Qt::RoundCap

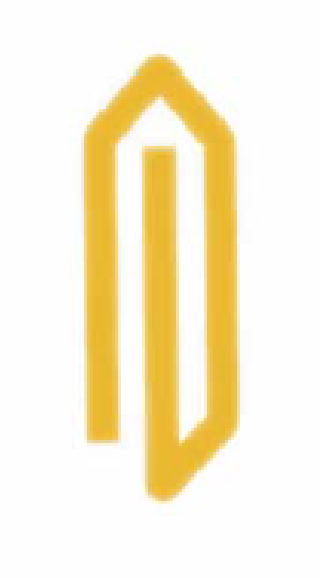
QPen



Qt::BevelJoin



Qt::MiterJoin



Qt::RoundJoin

QPen

```
void setColor ( const QColor & color )
void setStyle ( Qt::PenStyle style )
void setWidth ( int width )
void setWidthF ( qreal width )
Qt::PenStyle style () const
int width () const
qreal widthF () const
void setBrush ( const QBrush & brush )
void setCapStyle ( Qt::PenCapStyle style )
void setJoinStyle ( Qt::PenJoinStyle style )
```

QColor

```
QColor ( int r, int g, int b, int a = 255 )
QColor ( QRgb color )
QColor ( const QString & name )
QColor ( const char * name )
QColor ( const QColor & color )
QColor ( Qt::GlobalColor color )
void setCmyk ( int c, int m, int y, int k, int a =
    255 )
void setHsv ( int h, int s, int v, int a = 255 )
void setRgb ( int r, int g, int b, int a = 255 )
```

QColor

enum

Qt::GlobalColor Qt::white
Qt::black Qt::red
Qt::darkRed Qt::green
Qt::darkGreen Qt::blue
Qt::darkBlue Qt::cyan
Qt::darkCyan Qt::magenta
Qt::darkMagenta Qt::yellow
Qt::darkYellow Qt::gray
Qt::darkGray Qt::lightGray

QBrush

```
QBrush ( Qt::GlobalColor color, Qt::BrushStyle
    style = Qt::SolidPattern )
const QColor & color () const
const QGradient * gradient () const
void setColor ( const QColor & color )
void setColor ( Qt::GlobalColor color )
void setStyle ( Qt::BrushStyle style )
void setTexture ( const QPixmap & pixmap )
void setTextureImage ( const QImage & image )
Qt::BrushStyle style () const
QPixmap texture () const
QImage textureImage () const
```

QPainter

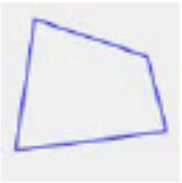
```
QPainter ( QPaintDevice * device )
bool begin ( QPaintDevice * device )
bool end ()
QPaintEngine * paintEngine () const
QPaintDevice * device () const
const QBrush & background () const
Qt::BGMode backgroundMode () const
const QBrush & brush () const
const QPen & pen () const
void setBackground ( const QBrush & brush )
void setBackgroundMode ( Qt::BGMode mode )
void setBrush ( const QBrush & brush )
void setBrush ( Qt::BrushStyle style )
void setPen ( const QPen & pen )
void setPen ( const QColor & color )
void setPen ( Qt::PenStyle style )
```

QPainter

```
void drawPoint ( const QPointF & position )
void drawLine ( const QLineF & line )
void drawLines ( const QLineF * lines, int
    lineCount )
void drawRect ( const QRectF & rectangle )
void drawRects ( const QRectF * rectangles, int
    rectCount )
void fillRect ( const QRectF & rectangle, const
    QBrush & brush )
void eraseRect ( const QRectF & rectangle )
```

QPainter

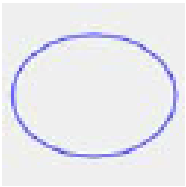
void drawRoundedRect (const QRectF & *rect*, qreal *xRadius*, qreal *yRadius*, Qt::SizeMode *mode* = Qt::AbsoluteSize)



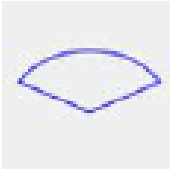
void drawPolygon (const QPolygonF & *points*, Qt::FillRule *fillRule* = Qt::OddEvenFill

void QPainter::drawPolyline (const QPointF * *points*, int *pointCount*)

QPainter

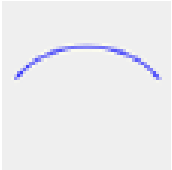


void drawEllipse (const QRectF & *rectangle*)

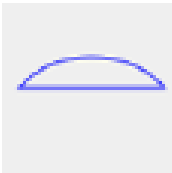


void drawPie (const QRectF & *rectangle*, int *startAngle*, int *spanAngle*)

QPainter



void drawArc (const QRectF & *rectangle*, int *startAngle*, int *spanAngle*)



void drawChord (const QRectF & *rectangle*, int *startAngle*, int *spanAngle*)

QPainter

```
void MyWidget::paintEvent(QPintEvent *event)
{
QPainter painter(this);
painter.setRenderHint(QPainter::Antialiasing, true)
    ;
painter.setPen(QPen(Qt::black, 12, Qt::DashDotLine,
     Qt::RoundCap));
painter.setBrush(QBrush(Qt::green, Qt::SolidPattern
    ));
painter.drawEllipse(80, 80, 400, 240);
QPointF points[4] = { QPointF(10.0, 80.0), QPointF
    (20.0, 10.0), QPointF(80.0, 30.0), QPointF
    (90.0, 70.0) };
painter.drawPolygon(points, 4);
QRectF rectangle(10.0, 20.0, 80.0, 60.0);
painter.drawRect(rectangle);
painter.drawText(100, 100, "Hello Qt");
painter.drawImage(300, 100, QImage(":/image.png"));
painter.drawImage(80, 80, 400, 240, 60 * 16, 270 *
```