QGraphicsItem: when paint function is called

I am creating an animation with QGraphicsView, QGraphicsScene and QGraphicsItem. Can someone explain me when the paint function is called? Although I does not change variables of the item, the paint function is called every ≈100 ms. Can I stop that, so i can repaint the item when I want?

qt graphics

asked Aug 30 '13 at 9:02

Maxii
207 ● 5 ● 21

2 Answers

You are approaching it the wrong way. The item should be repainted only when needed - when you change how it looks or where it's located. That's when you call the `QGraphicsItem::update()`. The rest will be handled for you. It seems you're overcomplicating things.

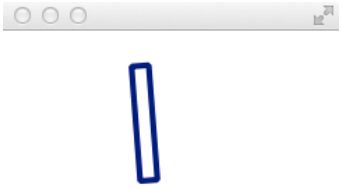
Do note that you need to be determining the current time-dependent parameter of the animation within the `paint()` method, or "close" to it (say, right before `update()` is called), using actual time! If your animations are derived from `QAbstractAnimation`, it's already done for you. If they are not, then you'll have to use `QElapsedTimer` yourself.

The relevant Qt documentation says:

> The animation framework calls `updateCurrentTime()` when current time has changed. By reimplementing this function, you can track the animation progress. Note that neither the interval between calls nor the number of calls to this function are defined; though, it will normally be 60 updates per second.

This means that Qt will do animations on a best-effort basis. The currentTime reported by the animation is the most recent time snapshot at the moment the animation was updated in the event loop. This is pretty much what you want.

The simplest way to deal with all this would be to use `QVariantAnimation` with `QGraphicsObject`. An example is below. Instead of rotating the object, you may have your own slot and modify it in some other way. You can also, instead of using signal-slot connection, have a customized `QVariantAnimation` that takes your custom `QGraphicsItem`-derived class as a target.



main.cpp

```
#include <QApplication>
#include <QGraphicsScene>
#include <QGraphicsView>
#include <QGraphicsObject>
#include <QPropertyAnimation>
#include <QGraphicsRectItem>

class EmptyGraphicsObject : public QGraphicsObject
{
public:
    EmptyGraphicsObject() {}
    QRectF boundingRect() const { return QRectF(0, 0, 0, 0); }
    void paint(QPainter *, const QStyleOptionGraphicsItem *, QWidget *) {}
};

class View : public QGraphicsView
{
public:
    View(QGraphicsScene *scene, QWidget *parent = 0) : QGraphicsView(scene, parent) {
        setRenderHint(QPainter::Antialiasing);
    }
    void resizeEvent(QResizeEvent *) {
        fitInView(-2, -2, 4, 4, Qt::KeepAspectRatio);
    }
};

void setupScene(QGraphicsScene &s)
{
    QGraphicsObject * obj = new EmptyGraphicsObject;
    QGraphicsRectItem * rect = new QGraphicsRectItem(-1, 0.3, 2, 0.3, obj);
    QPropertyAnimation * anim = new QPropertyAnimation(obj, "rotation", &s);
    s.addItem(obj);
    rect->setPen(QPen(Qt::darkBlue, 0.1));
    anim->setDuration(2000);
    anim->setStartValue(0);
    anim->setEndValue(360);
    anim->setEasingCurve(QEasingCurve::InBounce);
    anim->setLoopCount(-1);
    anim->start();
}

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    QGraphicsScene s;
    setupScene(s);
    View v(&s);
    v.show();
    return a.exec();
}
```

edited Aug 30 '13 at 18:36

answered Aug 30 '13 at 17:58

Kuba Ober

50.3k ● 7 ● 49 ● 108

You can set the viewportUpdateMode of the QGraphicsView to change how it updates. The options are: -

- QGraphicsView::FullViewportUpdate
- QGraphicsView::MinimalViewportUpdate
- QGraphicsView::SmartViewportUpdate
- QGraphicsView::BoundingRectViewportUpdate

- QGraphicsView::NoViewportUpdate

The Qt docs explains what the different options do, but if you want full control, just set to QGraphicsView::NoViewportUpdate and control it yourself using a QTimer event.

answered Aug 30 '13 at 9:47

TheDarkKnight
17.7k ● 1 ● 19 ● 50