

# Archived:Animating graphics item fading



Archived: This article is **archived** because it is not considered relevant for third-party developers creating commercial solutions today. If you think this article is still relevant, let us know by adding the template {{ReviewForRemovalFromArchive|user~~~~|write your reason here}}.

Qt Quick should be used for all UI development on mobile devices. The approach described in this article (using C++ for the Qt app UI) is deprecated.

## Overview

This series of snippets demonstrates how to implement nice animation effects into your Qt UI application. Qt 4.6 has Qt Animation Framework as a built-in feature. Documentation can be found here: [1] ↗

This snippet shows how to animate a fade effect in QGraphicsPixmapItem.

See full example code here: [File:Animation fw.zip](#)

## Preconditions

Qt 4.6 has been installed.

## Header file

Define fade property and setter/getter function.

```
class PictureItem: public QObject, public QGraphicsPixmapItem
{
    Q_OBJECT

    // Fade picture property for QPropertyAnimation
    Q_PROPERTY(int fade READ fade WRITE setFade)

public:
    // For fade property
    void setFade(int f);
    int fade() const;

    // Animate fade effects for this class
    void animateFadeIn();
    void animateFadeOut();

private:
    int m_fade;
    QPixmap m_originalPicForFade;
};
```

## Source file

Implement setter/getter for fade property.

```
void PictureItem::setFade(int f)
{
    m_fade = f;

    // Store original picture before fading it "broken"
    if ((m_fade == 255 || m_fade == 0) && m_originalPicForFade.isNull()) {
        m_originalPicForFade = pixmap();
    }

    // Picture for fading
    QPixmap pic;
    if (m_originalPicForFade.isNull())
        pic = pixmap();
    else
        pic = m_originalPicForFade;

    // Actual fading
    QPixmap transparent(pic.size());
    transparent.fill(Qt::transparent);
    QPainter p(&transparent);
    p.setCompositionMode(QPainter::CompositionMode_Source);
    p.drawPixmap(0, 0, pic);
    p.setCompositionMode(QPainter::CompositionMode_DestinationIn);
    p.fillRect(transparent.rect(), QColor(0, 0, 0, m_fade));
    p.end();
    pic = transparent;
    transparent = 0;

    // Use new faded picture
    setPixmap(pic);
}

int PictureItem::fade() const
{
    return m_fade;
}
```

The animation is executed from the `animateFadeIn()` or `animateFadeOut()` method.

```
void PictureItem::animateFadeIn()
{
    // Start animate this class
    QPropertyAnimation* anim = new QPropertyAnimation(this, "fade");
    anim->setDuration(1000);
    anim->setStartValue(255); // from fully visible
    anim->setEndValue(0); // to invisible
    anim->setEasingCurve(QEasingCurve::Linear);

    QObject::connect(anim, SIGNAL(finished()), this, SLOT(animationFinished()));

    anim->start(QAbstractAnimation::DeleteWhenStopped);
}

void PictureItem::animateFadeOut()
{
```

```
// Start animate this class
QPropertyAnimation* anim = new QPropertyAnimation(this, "fade");
anim->setDuration(1000);
anim->setStartValue(0); // from invisible
anim->setEndValue(255); // to fully visible
anim->setEasingCurve(QEasingCurve::Linear);

QObject::connect(anim, SIGNAL(finished()), this, SLOT(animationFinished()));

anim->start(QAbstractAnimation::DeleteWhenStopped);
}
```

## Postconditions

---

PictureItem can be faded by calling the PictureItem::animateFadeIn() or PictureItem::animateFadeOut() method.

## Animation Framework snippet series

---

### Full example code package

- Enabling Qt Animation Framework in an application
- Archived:Animating graphics item position in Qt
- Archived:Animating graphics item rotation
- **Archived:Animating graphics item fading**
- Archived:Animating graphics item scaling
- Archived:Animating graphics item position and rotation simultaneously