Archived:Animated Bubbles with QPainter

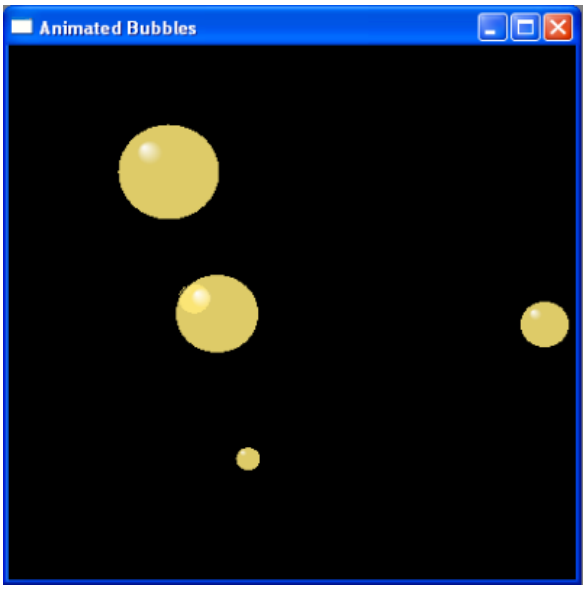
Archived: This article is archived because it is not considered relevant for third-party developers creating commercial solutions today. If you think this article is still relevant, let us know by adding the template {{ReviewForRemovalFromArchive|user=~~~~|*write your reason here*}}.

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.

This article shows how QPainter can be used to paint a scene rendered using OpenGL in a QGLWidget. Note that this example only runs on Windows Desktop.

Overview

OpenGL is being integrated in QGLWidget widget. QGLWidget also behaves like other standard Qt widget which supports widget properties also. Usually, QGLWidget is subclassed to display a pure 3D scene. In this example we will create translucent 2D bubbles.



Project File (.pro)

Add **opengl** to the project file

```
QT += opengl
```

Class Definition

```
#ifndef BUBBLESQT_H
#define BUBBLESQT_H

#include <QMainWindow>
#include <QGLWidget>
#include <QtGui>
#include <QtOpenGL>
#include "bubble.h"
#include <QTimer>
class Bubble;
class QPaintEvent;
class QWidget;
namespace Ui {
    class BubblesQt;
}
class BubblesQt : public QGLWidget
{
    Q_OBJECT
public:
    explicit BubblesQt(QWidget *parent = 0);
    ~BubblesQt();
 QSize sizeHint() const;
private:
    Ui::BubblesQt *ui;
protected:
    void paintEvent(QPaintEvent *event);
    void showEvent(QShowEvent *event);
private:
    void createBubbles(int number);
    QList<Bubble*> bubbles;
    QTimer animationTimer;
private slots:
   void animate();
};
```


(C) Copyright Nokia 2013. All rights reserved.

```
#endif // BUBBLESQT_H
```

The **BubblesQt** class inherits from QGLWidget. We reimplement the `paintEvent()` function to update the drawing area. `showEvent()` is used to initialise the 2D graphics used. The `animate()` slot is called periodically by the *animationTimer* to update the widget; the `createBubbles()` function initializes the bubbles.

Class Implementation

```
BubblesQt::BubblesQt(QWidget *parent) :
    QGLWidget( parent)
{
    animationTimer.setSingleShot(false);
    connect(&animationTimer, SIGNAL(timeout()), this, SLOT(animate()));
    animationTimer.start(25);
    setAutoFillBackground(false);
    setMinimumSize(200, 200);
    setWindowTitle(tr("Animated Bubbles"));
}
```

In the constructor, a timer is initialize to control the animation. `setAutoFillBackground()` is set to *false* to instruct `OpenGL` not to paint a background for the widget.

```
    QPainter painter(this);
    painter.setRenderHint(QPainter::Antialiasing);
    foreach (Bubble *bubble, bubbles) {
        if (bubble->rect().intersects(event->rect()))
            bubble->drawBubble(&painter);
    }
    painter.end();
```

We construct a QPainter to paint the widget with 2D graphics; in this case, we will draw a number of translucent bubbles onto the widget. When `QPainter::end()` is called, suitable OpenGL-specific calls are made to write the scene, and its additional contents, onto the widget.

```
void BubblesQt::showEvent(QShowEvent *event)
{
    Q_UNUSED(event);
    createBubbles(5);
}
```

`showEvent()` can be reimplemented in a subclass to receive widget show events which are passed in the event parameter.

```
void BubblesQt::animate()
{
    QMutableListIterator<Bubble*> iter(bubbles);
    while (iter.hasNext()) {
        Bubble *bubble = iter.next();
        bubble->move(rect());
    }
    update();
}
```

The `animate()` slot is called every time the widget's `animationTimer` emits the `timeout()` signal. This keeps the bubbles moving around.We simply iterate over the bubbles in the bubbles list, updating the widget before and after each of them is moved.

```
void BubblesQt::createBubbles(int number)
{
    for (int i = 0; i < number; ++i) {
        QPointF position(width()*(0.1 + (0.8*qrand()/(RAND_MAX+1.0))),
                         height()*(0.1 + (0.8*qrand()/(RAND_MAX+1.0))));
        qreal radius = qMin(width(), height())*(0.0125 + 0.0875*qrand()/(RAND_MAX+1.0));
        QPointF velocity(width()*0.0125*(-0.5 + qrand()/(RAND_MAX+1.0)),
                         height()*0.0125*(-0.5 + qrand()/(RAND_MAX+1.0)));
        bubbles.append(new Bubble(position, radius, velocity));
    }
}
```

`createBubbles()` is used to create number of bubbles in the scene.

Source Code

The full source code presented in this article is available here: File:BubblesQt.zip

Related Article

OpenGL with Qt

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.