Archived:How to apply transformation matrix to graphics item

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 (based on QGraphicsView) is deprecated for mobile devices.

This article shows how to use Transformation marix to graphics item.

Overview

Applying 2D transformations to GraphicsItems, GraphicsViews or directly to QPainter could be done setting a QTransform matrix to one of the class mentioned before.

The QTransform matrix is described in detail in the reference documentation. It replaces QMatrix which doesn't allow prospective transformations.

To apply a transform matrix is pretty easy, but when dealing with GraphicsItems in the same scene, you need to take more care of the transformations applied to the item's coordinate system to avoid undesired behaviors. The media player is loading...

Code

The video above shows an application with 2 graphic items: the red dot created with `QGraphicsScene::addEllipse` and a custom graphicsItem which makes use of QTransform. The custom graphics item re-implements the `QGraphicsViewItem::paint` method in order to draw itself in a "transformed" coordinate system. To keep the code simple and generic, QTransform matrix has not been modified and it's an identity matrix, as defined by the QTransform constructor.

```
QTransform t;
painter->setTransform(t);
```

As you can see from the video, at the beginning everything seems to work fine and the red dot is on the top-left corner of the custom item (white box), but when the window is resized (which happens on the mobile devices when the application changes orientation mode) the world coordinate system changes. In the above snippet, `QPainter::setTransform` overrides the world matrix which prevents the translation of the white box.

This underhand issue is pretty popular but it's not described elsewhere and this is why this Wiki page has been written. This problem happens because the `setTransform` method overwrites the world matrix by default whereas, in most cases, we want to combine our matrix with the world's one. The following code shows how to combine the two matrices.

```
QTransform t;
painter->setTransform(t, true);
```

More on Transformation

- Transformations with QWidgets
- Animation with Transformation in Qt