سوف تتعرف في هذ الفصل على الصفوف التالية ومايندرج تحتها
QgraphicsView و QGraphicsScene و QGraphicsItem و QGraphicsItemAnimation
وجميعها حديثة لأنها تم اصدارها في Qt4.2 وتندرج تحتهم عدة صفوف تسمح لك بزيادة من التخطيط والسهولة
وبإمكانك صنع اشكال جديد بالوراثة من QGraphicsItem عموما سوف نتحدث عن هذا الموضوع في وقت لاحق من هذا الملف .

وثم وفي الاصدار Qt4.4 تم تطوير هذا الموضوع ليتم اضافة QWidgetItem و layout خاص بها لتسهل عليكم عمل برامج عملاقة بأقل جهد.

وسوف نتحدث في هذا الملف عن QMatrix و QTransform
وكلاهما مهمين والفرق بينهما ان QMatrix مصفوفة ذات بعد ثالث و QTransForm اكبر وبياناتها اكثر ودقتها اعلى فلذلك التعامل معها يزداد صعوبة وكلاهما يفيد في تعيين نظام الاحداثيات فمثلا النقطة (0,0) تجدها في منتصف الصفحة بينما هي يجب ان تكون في الجانب الاعلى الأيسر على العموم سنتحدث ايضا عنها في هذا الملف .

QGraphicsView :

وهو الوسيلة الوحيد (viewport) لمشاهدة QGraphicsItem ولا يتم تركيب الأخير مباشرة عليها بل يجب ان يكون هناك وسيط بين QGraphicsItem و QGraphicsView وهو QGraphicsScene فيجب ان تضيف العناصر الى ال scene ثم اضافة ال scene الى viewport وهذا كنبدة عامة بسيطة عنها .
ومن خواص الكلاسس انه يرث من QScrollArea أي ان ال scene لو كبر حجمه (بالبكسل)عن الشاشة فإنه اوتوماتيك يتم اضهار ال scrollbar واليكم بعض الخواص والدوال
1- خواص السحب

enum QGraphicsView::DragMode

setDragMode ويمكن اضافتها باستخدام الدالة .

Constant	Value	Description
QGraphicsView::NoDrag	0	لا شيئ
QGraphicsView::ScrollHandDrag	1	ويوفر لك مقبض للتحكم على شكل يد للتجول في الصفحة والسحب فيها
QGraphicsView::RubberBandDrag	2	ويوفر لك القدر على تحديد الصور والتحديد على مجموعة صور

2- التعامل مع ال cache
الكمبيوترات الحديثة جعلت سعة ال كاش كبيرة جدا وذلك لهدف تسريع عمل البرامج ان سرعة الكاش في نقل النصوص والصور و... يوازي سرعة المعالج وفائدة الكاش هنا انه عند التنقل في الرسم او النص فإنه يتم تحميل النص او الصور من جديد كلما غابت وظهرت والكاش هو الوسيلة الوحيدة لتوفر لتوفير
num QGraphicsView::CacheModeFlag

setCacheMode ويمكن التعامل معها باستدعاء الدالة

الثابت	Value	الوصف
QGraphicsView::CacheNone	0x0	الرسم مباشرة يتم viewPort تحميله على ال
QGraphicsView::CacheBackground	0x1	هذا الخيار يؤثر على QBrush الخلفية التي تعتمد على او الخلفية التي تقوم QPainter انت برسمها بواسطة وهي تقوم بتحديد pixmap واحد يغطي مساحة ال viewport

3- update
عملية التحميل للمشهد تتم في كل مرة يتم فيها تغيير حجم widget او التنقل به او بداخله او استدعاء الدالة update ويمكنك ان تحدد الطريقة التي يحدث بها التحميل حسب رغبتك بالسرعة واستغلالك للذاكرة

enum QGraphicsView::ViewportUpdateMode

setViewportUpdateMode

Constant	Value	Description
QGraphicsView::FullViewportUpdate	0	ويتم فيها تحميل الرسم بالكامل وهي اسرع في المعالجة وتصرف وقت اكثر في اعادة الرسم ويفضل استخدامها مع الرسومات الصغيرة او الرسومات التي لاتدعم اعادة الرسم الجزئي مثل QGlWidget
QGraphicsView::MinimalViewportUpdate	1	وهي النمط الأساسي يغير فقط في المنطقة التي تم كشفها والباقي يدعه مثل ماكان عليه
QGraphicsView::SmartViewportUpdate	2	يحاول ايجاد نمط تحميل مثالي بحيث يعيد رسم التغير فقط طبعا ابطأ في المعالج وهو الأسرع في الرسم ويفيدك في الرسومات المتكررة المنطقية
QGraphicsView::BoundingRectViewportUpdate	4	يصنع ستطيل حول مناطق التغير ويعيد الرسم عليها
QGraphicsView::NoViewportUpdate	3	مفيدة فقط في الاطارات الثابتة بحيث لا يحدث تحميل اذا حدث أي تغيير

اختر الطريقة المناسبة للعرض او اجعله على نمط default لانه يجمع بين الحفاظ على سرعة المعالجة وتوفير بعض الوقت في اعادة الرسم وهناك دوال اخرى تفيدك وهي

itemAt يعيد مؤشر للعنصر عند نقطة محدد.
items وتعيد list ل مؤشرات العناصر الموجدة او يشملها شكل انت رسمته وسوف تجد مثال لاستخدامها في لعبة انصدام الفأر .
setBackgroundBrush لوضع خلفية بواسطة QBrush.
render هذه الدالة تخزن الرسم من scene الى QPainter المرسل في الوسيط الأول والوسيط الثاني يستقبل QRect وهو المساحة التي تريد الارسال اليها والوسيط الثالث يمثل المساحة التي يستقبل منها الشكل والأخير يحدد التناسب بينهم .
setMatrix و setTransform ويمكنك من خلالها ترسل QTransform or QMatrix.
sceneRect ترجع مساحة المشهد الذي تم تحديده في QGraphicsScene.
setInteractive وتستقبل قيمة بولينية اذا كانت true فإن المشهد يتفاعل مع الفارة والكيبورد مثلها مثل read only.
centerOn يتم تحريك السكرول بار الى النقطة التي تحددها .
setScene لإضافة QGraphicsScene
slots:
invalidateScene يقوم بإعادة رسم المساحة المحددة والطبقة المحددة .
updateScene يقوم بإعادة رسم مساحات عديدة تحدد بواسطة <QList<QRect> .
updateSceneRect اعادة الرسم في بقعة محددة .
والثلاثة الأخيرة يمكن استدعائها عند ضهور هذا ال signal
sceneRectChanged وهو يرسل المساحة التي تم فيها التغيير او changed وهو يرسل مجموعة المساحات التي تم عليها التغيير وجميعها م QGraphicsScene .
هناك دوال اخرى كثيرة يمكن مراجعة qt assistant

QGraphicsScene
وهي حاوية ل QGraphicsItem حيث لا يمكن اضافة الأخير مباشرة الى QGraphicsView بل يجب ان يمر من QGraphicsScene
لن نطيل الحديث عنه ونترككم مع اهم دواله
جميع الدوال التي تبدأ ب add تقوم بإضافة عنصر داخل المشهد وتعيد مؤشر له
collidingItems وتعيد لسته للعناصر المتصادمة او المتداخلة
setSelectionArea لوضع مساحة يتم فيها اختيار مجموع العناصر المحاط عليها وكأن تحيطها بالفأرة .
selectedItems ويرجع العناصر التي ت احاطتها بالفأرة او بالدالة السابقة
selectionArea يرجع مساحة العناصر المختارة
setFocus يضع التركي على عنصر ما كأنك ضغط عليه بالفأرة .
focusItem يرجع العنصر الذي عليه التركيز .
createItemGroup ينشئ group من لستة من العناصر حيث يربطهم رابط خفي اذا تحرك واحد منهم تحرك الآخرين معه وهلم جرا .
destroyItemGroup يدمر الرابط الخفي .
اما دوال البحث فهي
setItemIndexMethod وهي تنظم العناصر لتتيح لك البحث السريع بأحد الدوال التي سوف يأتي ذكرها لاحقا (items() و itemAt()) **وتكون اكثر قوة عندما تستخدمها للبحث في مشاهد ثابتة (غير متحركة)**
وهي تأخذ احد القيمتين:-
1-اذا كنت تستخدم مشاهد ثابتة واردت ان تسرع عملية البحث فأستخدم معها QGraphicsScene::BspTreeIndex (وهو يكون في الحالة الافتراضية لدى لا ضرورة لوضعها)
2- **اما اذا كان المشهد ذو حركة سريعة وخفت ان يؤثر الفرز على سرعة المشهد يمكن استخدام**
QGraphicsScene::NoIndex
setBspTreeDepth اذا كان نمط البحث =
QGraphicsScene::NoIndex
فلا داعي من استخدامك لهذه الدالة لأنها تستهلك من الذاكرة وتؤثر على الأداء وكل مازاد العمق زاد البطء والعدد المثال هو مابين 0 الى 10 عناصر في العمق الواحد فلو لديك 100 عنصر فيفضل وضع الرقم 10 داخل الوسيط ليصبح لكل قسم عشرة عناصر .
عموما اعرف ماتريده تم اختر
معلومة: BSP تعني تقسيم المشهد الثنائي وهي خوارزمية تساعد على الفهرسة لعملية البحث الثنائي .
itemAt ويعيد العنصر في النقطة التي قمت بتحديدها

items يعيد لستة من العناصر التي تحيط بها المساحة المدخلة
:Slots
clear يحذف جميع العناصر داخل المشهد
clearSelection يحذف العناصر المختارة داخل المشهد
advance وتعمل على تحريك المشهد لكل عنصر على العموم
سوف تعرف عملها بالتحديد حين نتحدث عن
QGraphicsItemAnimation

invalidate وهو شبيه لعمل نظيره في QGraphicsView.
update وهو شبيه لعمل نظيره في QGraphicsView.
: Signals
changed وتعيد مجموعة المساحات التي تم في التغيير سوء بحركة او غيره.
sceneRectChanged وتعيد مستطيل واحد يحوي جميع التغيرات.
selectionChanged اذا تم تغيير العناصر المختارة بواسطة اللإارة او دوال اخرى فإنها ترسل اشارة بدون ان تعيد شيئ .

وهناك دوال محمية اخرى تدعم الأحداث لا نستطيع التكلم عنها هنا ولكن سوف نتحدث عنها في الأمثلة القادمة لأن كل حدث يمثل صف بأكمله .

QGraphicsItem

وهو الكائن الذي كنا نتكلم عنه في جميع حديثنا السابق ويوجد منه عدة اشكال
ترث منه وهي على التوالي

الصف	فائدته
QGraphicsItemGroup	لصنع مجموعة من العناصر يرأسهم عنصر واحد .
QGraphicsPixmapItem	لصنع صورة داخل المشهد.
QGraphicsRectItem	لصن مستطيل داخل المشهد.
QGraphicsTextItem	لصنع صندوق نص معقد(مثله مثل QTextEdit).
QAbstractGraphicsShapeItem	وهو صف موروث من صفوف اخرى ليوفر لهم QBrus & QPen
QGraphicsEllipseItem	لرسم شكل بيضاوي
QGraphicsPathItem	لرسم QPainterPath
QGraphicsPolygonItem	لرسم شكل متعدد الأضلاع
QGraphicsSimpleTextItem	لرسم مربع نص بسيط(ليس به رسومات وجداول و....)
QGraphicsLineItem	لرسم خط

تذكر جميع هذه الصفوف لكي تظهر يجب اضافتها الى scene ومن ثم الى graphics view
جميع الدوال في المصفوفات السابقة يمكن تعلمها بسهولة لعدم توفر فيها أي صعوبة او شيئ ديد يذكر .

الدوال المهمة في QGraphicsItem

1-الاحداثيات

- **x** ويرجع الحداث السيني للعنصر.
- **y** ويرجع الاحداث الصادي للعنصر.

zValue ليس المقصود هنا ان الشكل ثلاثي الأبعاد لنأخذ مثالا اذا كان لديك 3 عناصر ووضعتهم فوق بعضم فإن الذي في الأسف قيمته 0 والذي في الوسط قيمته 1 والذي في الألى قيمته 2 فهو يحدد مستوى ارتفاع الطبقة فلو لديك عنصر قيمة ال z لديه 100 وعنصر اخ قيمته 30 فإنك تجد العنصر الذي قيمته 100 فوق العنصر الذي قيمته 30
ويمكن تحديد القيمة بالدالة setZValue .
pos ويعيد نقطة الموقع للعنصر.
setPos ويحدد موقع العنصر .

2- اخرى

setFocus ويضع التركيز على العنصر كأنك ضغط عليه بالفأر.
setEnabled ان كانت القيمة false فلا داعي ان تتعب نفسك بالضغط عليه .

setData وهي قيم يحتفظبها العنصر بداخله لكي تستطيع استدعئها في وقت لاحق على شكل QVarian .
setCursor تغيير شكل الفأرة للعنصر .
moveBy تحريك العنصر من موقعه السابق + النقطة المطلوبة فكأنك تقول تحرك كذا خطوة في الاحداثي السيني وكذا خطوة في الأحداثي الصادي .
جميع الدوال التي تبدأ ب mapTo فإنها تقوم بإرجاع الشكل الذي تريده مع نسخ ال نظام الحداثيات عليها وهي بذلك تعتبر من أهم الدوال في تسهيل تعيين الاحداثيات
ويعكسها الدوال التي تبدأ ب mapFrom فإنه يرجع الشكل الذي تريده مع نسخ عكس نظام الاحداثيات على العنصر المرجع .
boundingRect ويعيد مستطيل يحيط بالعنصر ويبقى حوله .
advance اذا كان للعنصر حركة فهي تقوم بالانتقال الى خط الزمن اذي تحدده انت في الوسيط.
grabMouse وعكسها ungrabMouse وفيها يتلقى العصر جميع احداث الفأرة التي تحدث في المشهد .
grabKeyboard وعكسها ungrabKeyboard وفيها العنصر يتلقى جميع الأحداث التي تحدث في المشهد.
وهناك دوال اخرى عديدة لمزيد من المعلومات راجع ملف Qt assistant
paint ان اردت انت تشئ شكل جديد غير الأشكال المتوفرة فما عليك سوى ان ترث هذا الكلاسس وتبدأ بالرسم عليه بواسط QPainter .
installSceneEventFilter يمكن بهذا الأمر انت تفوض عنصر يتلقى احداث هذا العنصر ويتم معالجته بواسة SceneEventFilter
setFlags

الثابت	قيمته	شرح
QGraphicsItem::ItemIsMovable	0x1	يمكنك من تحريك العنصر عن طريق الفارة
QGraphicsItem::ItemIsSelectable	0x2	يمكن من اختيار العنصر سواءء بالفأرة او بأحد دوال الإختيار
QGraphicsItem::ItemIsFocusable	0x4	يمكنك من التركيز على عنصر سواء بواسطة الفار او بواسطة setFocus
QGraphicsItem::ItemClipsToShape	0x8	يتم اخفاء أي رسم غير مرسوم فوقها او جاخفاء الجزء الغير مرسوم فوق هذا العنصر
QGraphicsItem::ItemClipsChildrenToShape	0x10	إن كان العنصر أب لمجموعة فإن أبناءه لا يظهر منهم إلا الجزء

المرسوم فوقه		
لاتخف من نظزام الإحداثيات ثق تمام انه سوف يتم تجاهل أي تغير لم يضف الى العنصر وان رسم العنصر بعدها هذا ان وضعت هذا الخيار للعنصر.	0x20	QGraphicsItem::ItemIgnoresTransformations

ويمكن استخدام الدالة setFlag لتحديد الخيار وقيمته البولينية بالرفض او القبول ان كانت القيمة true.

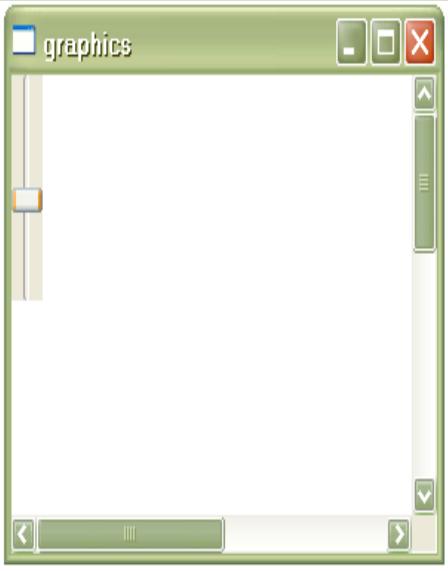
دوال التأكيد

isEnabled	**اذا كان العنصر لا يتجاوب مع الأحداث التي ذكرناها في الدالة setEnable فإن الدالة ترجع true**
isObscured	**تعيد القيمة true اذا غطي العنصر بالكامل من قبل عنصر آخر عتم .**
isObscuredBy	**مثل سابقتها ولكن بإمكانك اختبار العنصر الذي غطا على العنصر الحالي ان كان هو اعاد القيمة true**
isVisible	**اذا كان الجسم مرئي فسوف يعيد true**
isVisibleTo	**اذا كان العنصر لايرى الى اذا كان داخل العنصر الأب وارسلت للدالة مؤشر للأب فإنها تعيد true**
isWidget	**اذا كان العنصر widget فإنها تعيد true**
isWindow	**اذا كان العنصر نافذة تعيد true والفرق بين window & widget ان الأخير يتميز بإطار يحوي عنوان بغض النظر ان كان اب او ابن .**
isAncestorOf	**يعيد trueاذا كان العنصر اب للطفل المرسل مؤشره او جد او مافوقه**
isUnderMouse	**يعيد true اذا كان العنصر تحت مؤشر الفأرة**

QGraphicsWidget

هل رأيت كيف يعمل برنامج designer على Qt يتبادر في دهنك مالذي يجعل هذه النوافذ تتحرك بهذه السهولة الجواب هنا هذا الكلاسس يجمع بين الجرافيكس والنوافذ لتستطيع مستقبلا ان تحاكي برامج مثل الفلاش اعتقد بعد هذه الثورة في Qt4.4 سوف يزيد عدد مستخدمي المكتبة الا الضعف وأكثر انظر الى الصورة في الشكل (1)

```
#include <QApplication>
#include <QtGui>
//
int main(int argc, char ** argv)
{
    QApplication app( argc, argv );
    QGraphicsView view;
    QGraphicsScene scene(0,0,500,500);
    QSlider slider;
    QGraphicsProxyWidget sliderG;
    sliderG.setWidget(&slider);
    scene.addItem(&sliderG);
    view.setScene(&scene);
    view.show();
    return app.exec();
}
```



الشكل 1

كما تلاحظ في الشكل(1) فإنه تم دمج ال widget في المشهد .

والكلاسس الوحيد الذي يرث من QGraphicsWidget هو QGraphicsProxyWidget

وهو الواجهة لاضافة نافذة وكما ترث QGraphicsWidget من عناصر ال layout وهي :

QGraphicsLayoutItem ويرث منه QGraphicsLayout و QGraphicsWidget

ويرث من QGraphicsGridLayout QGraphicsLayout و QGraphicsLinearLayout

مهم جدا : شجرة وراثة الصفوف يجب أن تحفظها .

وجميعها عملها ودوالها لا تختلف عن عمل ودوال QLayout و من يرث منها

وبما ان هذا الدرس لمتقدم في المستوى فيجب سلفا انك تعرف ان تتعامل مع QLayout

ومن يرث منها .

وجميع الدوال في QGraphicsWidget يجب ان تعرفها في QWidget في النهاية هي محاكاة للأخير ولكن على هيئة جرافيكس .

QGraphicsItemAnimation

اذا سألك صديقك هل تعرف للفلاش فقل له لا انا اعرف لل Qt .

انا هذا الصف يعطي لعملك الحركة التي تريدها فقط تحتاج لشيئين:

1-العنصر

وقد تحدثنا عنه مسبقا

2-خط الزمن

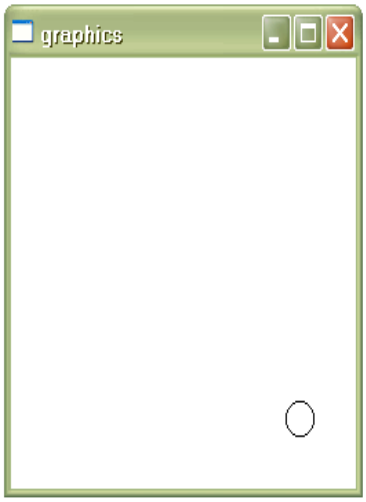
ويمكنك استدعاء الكلاسس QTimeLine وتتوفر به دوال ايقاف وتحريك المشهد

واضافة مدة العرض وعدد اطارات العرض(اطلع على time line في الفلاش او السويتش وتعرف جميع الدوال)

لقد تحدثنا في المشهد عن slots يدعى advance ووعدناك ان نقوم بشرحها حين يأتي وقتها هناك في الحقيقة عدة خطوط زمن يتم فيها التحرك الخطوط الفرعية وهي خطوط الزمن في QGraphicsItemAnimation

وخط عام وحيد ولاينتهي وهو خط الزمن التابع للمشهد ان تحرك تحركت معه جميع الخطوط الفرعية .

```
#include <QApplication>
#include <QtGui>
//
int main(int argc, char ** argv)
{
    QApplication app( argc, argv );
    QGraphicsItem *ball = new QGraphicsEllipseItem(0, 0, 20, 20);
    QTimeLine *timer = new QTimeLine(5000);
    timer->setFrameRange(0, 100);
    QGraphicsItemAnimation *animation = new QGraphicsItemAnimation;
    animation->setItem(ball);
    animation->setTimeLine(timer);
    for (int i = 0; i < 200; ++i)
        animation->setPosAt(i / 200.0, QPointF(i, i));
    QGraphicsScene *scene = new QGraphicsScene();
    scene->setSceneRect(0, 0, 250, 250);
    scene->addItem(ball);
    QGraphicsView *view = new QGraphicsView(scene);
    view->show();
    timer->setLoopCount(3);
    timer->start();
    return app.exec();
}
```



الشكل(2)

انظر الشكل (2) تجد اننا قمنا في بادئ الأمر قمنا بصنع العنصر ثم بصنع خط الزمن ثم استدعينا صف الحركة وقمنا بدمج خط الزمن مع العنصر ومن ثم قمنا بصنع الحركة حيث جعلنا الجسم يغير موقعه في كل اطار ليتحرك بشكل قطري وفي الأخير طلبنا اعادة المشهد 3 مرات جرب عمل مثل هذا الكود .

QMatrix & QTransform

وهما نظامي احداثيات يفيد في تعيين نظام الاحداثيات الجديد .
فالنقطة (0,0) ليس بالضرورة ان تكون في الطرف الأيسر في الأعلى
ونظام الاحداثيات الجديد يتكون من مصفوفة شكل(3).

M11	M12	0
M21	M22	0
dx	dy	1

شكل(3)

```
x' = m11*x + m21*y + dx
y' = m22*y + m12*x + dy
```

المعادلتين السابقتين تمثل الاحداثيات الجديدة فمثلا النقطة 0 و 0 اذا اردت ايجاد مكانهما الجديد يمكنك استخدام المعادلتين ولقد قامت كيو تي بإراحتك من عملية الحساب لتنتج لك Map ارسل لها نقطة وهي ترسل لك نقطة على النظام الذي صنعته انت او ارسل لها اي شيئ تريد مستطيل متعدد اضلاع ..... وهي ترسل الشكل الذي يقابله في نظام الاحداثيات الذي قمت بإنشائه .

كل جزء من المصفوفة يمثل رقم ولكن ماهي هذه الأرقام اليك الجدول التالي:

dx	**ويمثل إحداث النقل السيني (translate)**
dy	**ويمثل إحداث النقل الصادي (translate)**
M11	**ويمثل إحداث المقياس السيني(scale)**
M22	**ويمثل إحداث المقياس الصادي(scale)**
M12	**ويمثل إحداث القص الصادي(shear)**
M21	**ويمثل إحداث القص السيني(shear)**

وهناك دالة حسابية شبه مفيدة اخرا وهي det لتعطيك رقم يمثل محددة المصفوفة وهي في هذه الحالة محدة مصفوفة من الدرجة الثالثة .
ولكن اهم الدوال هنا هي Translat وهي للنقل و shear وهي للقص و rotate وهي للدوران و scale وهي للمساحة (المقياس).

الدالة inverted وهي تستقبل مؤشر الى متغير بوليني وتعيد القيمة وتعيد معكوس المصفوفة وترجع الى المؤشر القيمة true والا فإنها ترجع المصفوفة جميع عناصرها اصفار ماعدا ال scale Indentity matrix .

نصائح مهمة :

1-**توجد خاصية translate , rotate , scale ,shear في العناصر او المشهد وجميعها تأثر على بعضها .**

2-**ان وضع العنصر في المشهد قبل وضع أي من الخواص او الدوال السابقة فإنه لا يؤثر عليه لذلك من الأفضل وضع العناصر جميعها على المشهد أولا ثم التحكم بها عن طريق setMatrix او استدعاء أي من الدوال السابقة على العنصر او المشهد مباشرة .**

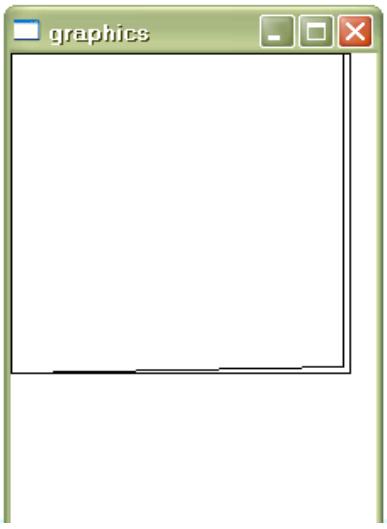
3-**يختلف transform عن matrix بعناصر يمكن حصرها في :**

m13 & m23 & m31 & m32 & m33

هذه الأرقام تزيد من دقة احداثيات النظام وتزيد من صعوبة الحسابات والتعامل معها ولكن كما ذكرنا سابقا ضع عناصرك على المشهد قبل ان تضيف matrix او transform على أي عنصر لأنها سوف تؤثر على بقية العناصر وبالتالي يؤثر على مسار عملك .

M13	**مرتسم هندسي على مستوى في محوره السيني**
M23	**مرتسم هندسي على مستوى في محوره الصادي**
M31	**dx**
M32	**dy**
M33	**وقيمته الأصلية 1 وكلما زادت القيمة صغر مقياس الرسم وهي تعني هنا كل عدد بكسل = واحد بكسل**

```
#include <QApplication>
#include <QtGui>
//
int main(int argc, char ** argv)
{
    QApplication app( argc, argv );
    QPixmap pixmap(400,400);
    pixmap.fill();
    QPainter painter(&pixmap);
    painter.drawRect(0,0,200,200);
    QTransform transform(1,0,.0001,0,1,0,0,0,1);
    //m13=.0001 and m11=1 for scale and m22=1
    painter.setTransform(transform);
    painter.drawRect(0,0,200,200);
    QLabel label;
    label.setPixmap(pixmap);
    label.show();
    return app.exec();
}
```



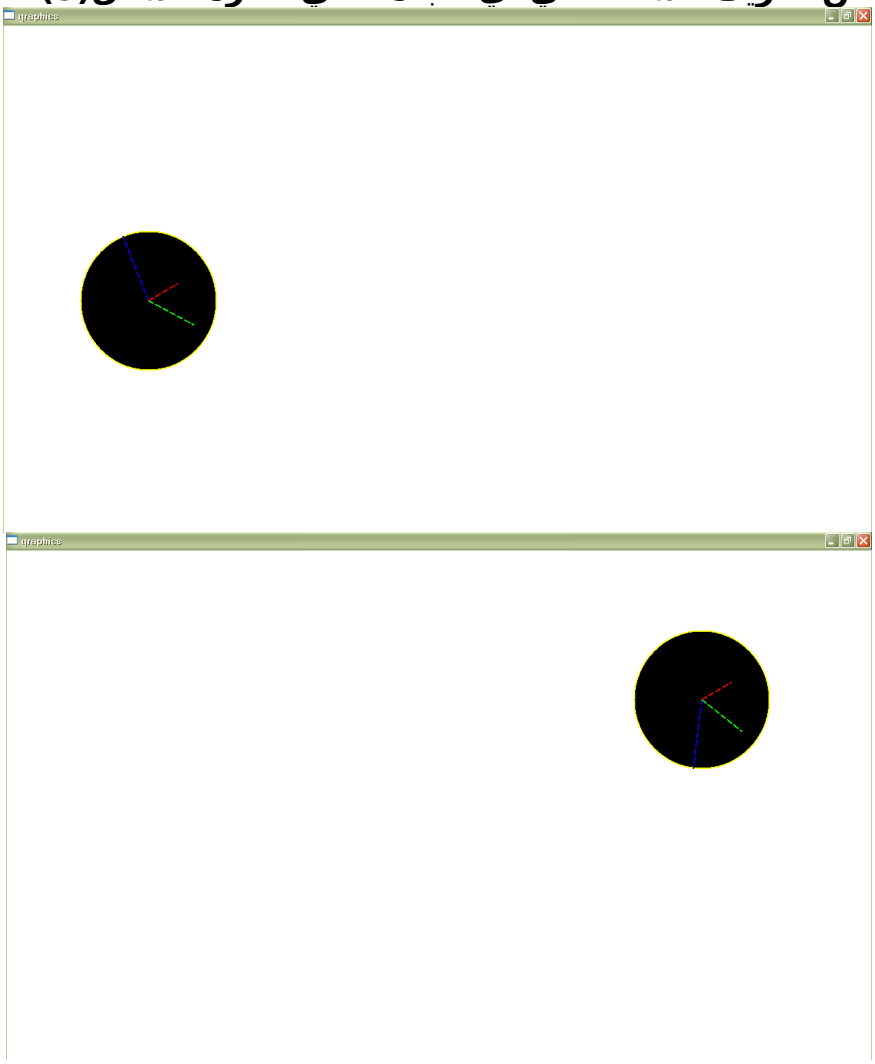
الشكل (4)

في الشكل (4) قمنا في البداية بإنشاء QPixmap لنرسم فوقها باستخدام QPainter بعد ان انشأنا لوحة الرسم ومقياسها 400 * 400 قمنا بتعبئتها بإستخدام الدالة fill وهي التي تأخذ اللون الأبيض الافتراضي مالم تقم انت بتغييره تم انشأنا الدالة QPainter

واعطينا مؤشر للوحة التي سوف نرسم فوقها وقمنا برسم مربع بإستخدام الدالة drawRect من النقطة 0و0 الى النقطة 200 و200 ثم بعد رسمها قمنا بإنشاء matrix باستخدا الكلاسس QTransform واضفناه للوحة الرسم ملاحظ المربع الأول لن يتأثر لأن قمنا برسمه قبل تغيير احداثيات النظام بينما لاحظ المربع الثاني وهو فوق المربع الأول فقط تم تغيير شكله قليلا ثم انشأنا lable جديد لنظيف اليه الرسم ثم قمنا بإظهاره .

الساعة

مثال الساعة :
سوف نقوم في هذا المثال بصنع ساعة عقارب يحوي عقرب الساعات وعقرب الدقائق وعقرب الثواني ويمكن تحريك الساعة في أي اتجاه القي نظرة للشكل(5) .



الشكل (5)

ان هذا العمل قد يأخذ معك الكثير في لغة اخرى ولكن مع c++ ومكتبة Qt فإنك محظوظ.
هذا العمل يحتوي على ثلاث ملفات
Clock.h & clock.cpp & main.cpp
وسوف نتكلم عن كل ملف على حذا مع شرح محتوياته.

clock.h -1

```
#ifndef __CLOCK_H__
#define __CLOCK_H__
#include <QGraphicsView>
#include <QtGui>
class cl:public QGraphicsView{
Q_OBJECT
public:
cl(QWidget * parent=0);
void steph();//تحرك عقرب الساعات خطوة
void stepm();//تحرك عقرب الدقائق خطوة
void steps();//تحرك عقرب الثواني خطوة
void shape();//ويقوم بتحديد شكل ولون العقارب
//=====================================
=====
private:
QGraphicsLineItem h;//عقرب الساعات
QGraphicsLineItem m;//عقرب الدقائق
QGraphicsLineItem s;//عقرب الثواني
QGraphicsEllipseItem back;//الدائرية  خلفية الساعة
QGraphicsItemGroup* group;//وتشمل شكل الساعة بالكامل
//=====================================
=====
protected:
void timerEvent ( QTimerEvent * event );//ويتلقى احداث ال المؤقت
}
;
class A{
};
#endif
```

في هذا الملف قمنا بإنشاء الصف cl وهو اختصار ل clock ويرث الصف من QGraphicsView .

clock.cpp -2
لو ألقيت نظرة الى ملف الرأس وتطلعت على الدوال ستجد التالي دوال للحركة ودوال الشكل العام للساعة ومتغيرات العقارب والخلفية ودالة تلقي احداث المؤقت ودالة البناء وسوف تحوي على اغلب الشفرة
1- دالة شكل الساعة

```
void cl::shape(){
h.setPen(QPen(QBrush(Qt::red),2,Qt::DashLine,Qt::RoundCap));
m.setPen(QPen(QBrush(Qt::green),2,Qt::DashLine,Qt::RoundCap));
s.setPen(QPen(QBrush(Qt::blue),2,Qt::DashLine,Qt::RoundCap));
h.setLine(0,0,0,-50);
m.setLine(0,0,0,-75);
s.setLine(0,0,0,-100);
back.setPen(QPen(QBrush(Qt::yellow),2));
back.setBrush(Qt::black);
back.setRect(-100,-100,200,200);
h.setZValue(3);
back.setZValue(0);
m.setZValue(2);
s.setZValue(1);
    }
```

لقد بدأنا بتحديد شكل العقرب ولونه لجميع العقارب ثم حددنا اطوالهم ومواقعهم فطول عقرب الثواني 100 بكسل وعقرب الدقائق 75 بكسل وعقرب الساعات 50 بكسل وتبدأ من النقطة 0 و 0 وتتحرك بالإتجاه السالب الصادي (للأعلى) لتشير الى الساعة 12 او الدقيقة والثانية 60 .
ثم انتقلنا الى رسم الخلفية وتحديد اطرافه ولونه الداخلي .
ملاحظة :في بداية العمل على المشهد يتم تحديد نظام الاحداثيات على النقطة 0 و 0 ولقد رسمنا الدائرة (الخلفية) لتكون النقطة 0و0 في وسطها فبذلك تدور عقارب الساعة من مركز الدائرة .
وفي الأخير قمنا بتعيين ارتفاع كل عنصر فجعلنا الخلفية في الأسفل وتأخذ القيمة 0 ومن ثم يعلوها عقارب الثواني وم ثم الدقائق ويعلوهم جميعا عقرب الساعات .
(لقد تم ذكر عمل zValue في حديثنا عن QGraphicsItem)

2-دوال حركة العقارب

```
void cl::steph(){
h.rotate(3);
}
//================================
void cl::stepm(){
m.rotate(.6);
}
//================================
void cl::steps(){
s.rotate(.6);
}
```

الأول يقوم بإدارة العقرب 3 درجات من اصل 360 درجة للدورة الكاملة
ملاحظة: القيم الموجبة تدير الشكل مع عقارب الساعة والسالبة عكس عقارب الساعة وهو عكس المتعارف عليه.

والثاني يقوم بتدوير عقرب الدقائق ما مقداره .6 درجة والثالث للثواني ايضا وبذات المقدار .

2-دالة البناء

```
cl::cl(QWidget *parent):QGraphicsView(parent){
QGraphicsScene *scene=new QGraphicsScene(0,0,200,200);
this->shape();
QList <QGraphicsItem*> list;
list<<&back<<&s<<&m<<&h;
group=(scene->createItemGroup(list));
this->setScene(scene);
group->setFlag(QGraphicsItem::ItemIsMovable,1);
QTime time=QTime::currentTime();
m.rotate(time.minute()*6);
h.rotate(time.hour()*30);
s.rotate(time.second()*6);
this->startTimer(100);
this->startTimer(6000);
this->startTimer(360000);
    }
```

تحتوي دالة البناء على ال scene حيث قمنا ببناء مؤشر له وتم بعد ذلك استدعاء الدالة shape() التي قمنا بصنعها لتحوي بناء شكل الساعة ثم قمنا بصناعة قائمة لكي نصنع منها group لكي تساعدنا في تحريكها عناصر الساعة سويا ثم اضفنا امكانية تحريك الساعة بواسطة الدالة

```
group->setFlag(QGraphicsItem::ItemIsMovable,1);
```

ثم قمنا بإستدعاء مصفوفة الوقت لكي نقوم بتحديد الوقت ومن تم تحديد اماكن عقارب الساعة المبدئية

وهو مافعلناه حقا في الأسطر التي تليها وفي النهاية قمنا بإنشاء مؤقتات وهي تستدعي الدالة

```
void timerEvent ( QTimerEvent * event );//ويتلقى احداث ال المؤقت
```

(من اللغة)كلما مر الزمن ثم يبدأ من جديد مالم يتم استدعاء الدالة

```
(QObject::killTimer(int id
```

حيث يعتبر كل 1000 ثانية واحدة .

3-دالة استقبال المؤقت

```
void cl::timerEvent ( QTimerEvent * event ){
if(event->timerId()==4)
steph();
if(event->timerId()==3)
stepm();
if(event->timerId()==2)
steps();
    }
```

عند انشائك لمؤقت فإنه يحتوي على id يتم ترقيمه من اول مؤقت الى الأخير بطريقة تسلسلية فقد قمنا بجع الأول للثواني والثاني للدقائق والثالث للساعات وكل مؤقت خاص بعقربه يبدأ من جديد فإن العقرب المطلوب يتم تحريكه .

3-الملف main.cpp

```
#include <QApplication>
#include "clock.h"
#include <QtGui>
#include <QtCore>
//
int main(int argc, char ** argv)
{
QApplication app( argc, argv );
cl a;
a.show();
return app.exec();
    }
```

قمنا في نهاية هذا العمل باستدعاء الصف الذي قمنا بصنعه ثم طلبنا إظهاره.
ملف clock.cpp كاملا.

```
#include <QtGui>
#include <QtCore>
#include "clock.h"
//================================
cl::cl(QWidget *parent):QGraphicsView(parent){
QGraphicsScene *scene=new QGraphicsScene(0,0,200,200);
this->shape();
QList <QGraphicsItem*> list;
list<<&back<<&s<<&m<<&h;
group=(scene->createItemGroup(list));
this->setScene(scene);
group->setFlag(QGraphicsItem::ItemIsMovable,1);
QTime time=QTime::currentTime();
m.rotate(time.minute()*6);
h.rotate(time.hour()*30);
s.rotate(time.second()*6);
this->startTimer(100);
this->startTimer(6000);
this->startTimer(360000);
}
//================================
void cl::shape(){
h.setPen(QPen(QBrush(Qt::red),2,Qt::DashLine,Qt::RoundCap));
m.setPen(QPen(QBrush(Qt::green),2,Qt::DashLine,Qt::RoundCap));
s.setPen(QPen(QBrush(Qt::blue),2,Qt::DashLine,Qt::RoundCap));
h.setLine(0,0,0,-50);
m.setLine(0,0,0,-75);
s.setLine(0,0,0,-100);
back.setPen(QPen(QBrush(Qt::yellow),2));
back.setBrush(Qt::black);
back.setRect(-100,-100,200,200);
h.setZValue(3);
back.setZValue(0);
```

```
m.setZValue(2);
s.setZValue(1);
}
//====================================
void cl::steph(){
h.rotate(3);
}
//====================================
void cl::stepm(){
m.rotate(.6);
}
//====================================
void cl::steps(){
s.rotate(.6);
}
//====================================
void cl::timerEvent ( QTimerEvent * event ){
if(event->timerId()==4)
steph();
if(event->timerId()==3)
stepm();
if(event->timerId()==2)
steps();
}
//====================================
```

في حال تواجد أي ملاحظة على المقال او تصحيح يمكن مراسلتي على البريد التالي mohammd.lbdl@gmail.com
سائلا المولى عز وجل ان يجعل ماكتبته في موازين حسناتي وان يوفقني وإياكم الى ما فيه رضاه وخدمة الآخرين.
تم الإنتها من هذا الباب بتاريخ 21 يونيو 2008.
اخوكم: محمد العبدلي .
رئيس فريق مصفوفة www.masfofa.com (هدفي عربي طموحي عربي) .

للإستفسار راسلنا على info@masfofa.com .
مشرف موقع www.qt-ar.com (طريقك نحو احتراف Qt) .