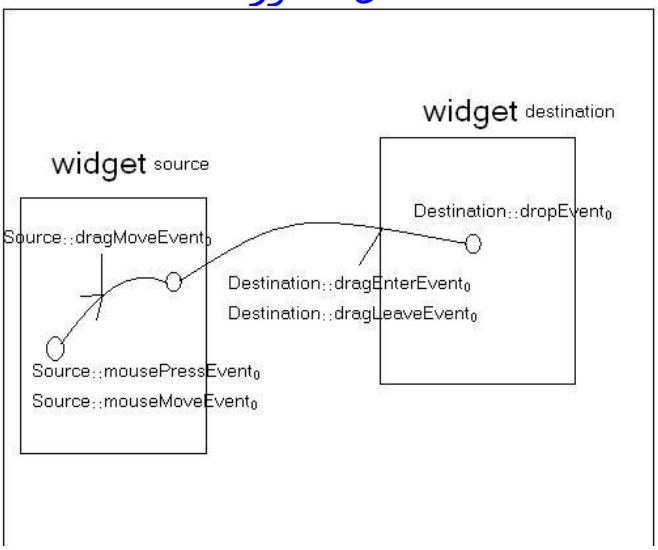
السحب والإفلات

ان هذه التقنية التي تراها في الألعاب وغيرها ماهي الا صورة بها بيانات يتم نقل البيانات من موضع الى آخر ويكون للصورة فيها مقبض تتمسك بموقع الفأرة وكغيرها من الأحداث فهي لها صفوف تديرها يقودها الصف Qdrag وصفوف الاحداث

الحدث	وصف
QDragEnterEvent	ويحدث عند الدخول على نافذة جديدة
QMoveEvent	ويحدث عند تحريك العنصر
QLeaveEvent	ويحدث عند مغادرته لنافذه
QDropEvent	ويحدث عند اسقاط العنصر على النافذة

مثال مصور



(1)

في الدوال السابقة جميعها كنا نتحدث عن سحب الصورة ودورة النقل التي سوف تمر فيها ولكن السؤال الآخر هنا اين هي هذه البيانات التي سوف يتم ارسالها مع الصورة ؟

الجواب هو لذا النوع MIME:

فهو يقوم بتخزان البيانات بالإضافة الى مفاتيح تعبر عن نوع البيانات بالطبع لك الحرية في تحديد نوعها بدون قيود لتحتاجها فيما بعد في استدعاء البيانات والصف الذي يقوم بهذا الدور هو QmimeData ويمكن اضافته بإستخدام الدالة setMimeData من الصف Qdrag لتضع بداخلها متغير من النوع QmimeData ولكي تستدعيها عليك بإستخدام الدالة mimeData من نفس الصف Qdrag او من الصف QdropEvent في حال الإسقاط للعنصر .

انت تعلمت كيف تجلب ال النوع mime وتضعه في عملية السحب والإفلات ولكنك لم تعرف كيف تقوم بتخزين البيانات وإستخراجه من الصف QmimeData ؟

قلنا في بادئ الأمر أن هناك مفاتيح و بيانات لذلك عند تخزين البيانت بواسطة الدالة setData
يجب أن تضع النوع في الوسيط الأول و البيانات في الوسيط الثاني
ليمكنك بعد ذلك من سحب البيانات بتحديد المفتاح بإستخدام الدالة data بعد وضعك المفتاح في وسيطها الأول والوحيد ويمكنك بوضع بعض البيانات مباشرة بدون الحاجة لتحديد النوع لأنها في الحقيقة تم تحديد نوعها مسبقا وهي:

الدالة	النوع (المفتاح)	شرح
setText	Text/plain	لتخزين النصوص البسيطة
setUrls	Text/uri-list	لتخزين متغيرات Qurl في حاوية QList
setImageData	----------	لتخزين متغيرات من النوع Qvariant
setHtml	Text/html	لتخزين نصوص html

وجميع هذه الأنواع يمكنك جلبها بعد تحديد النوع (المفتاح) او جلبها مباشرة بما يكافئها من دوال
مثل text و urls و..... , ويمكنك حذف البيانات التي لا تريدها بإستخدام removeFormat بوضع نوع(مفتاح) البيانات .

لقد أصبحت الآن تتعامل بشكل إحترافي مع نقل البيانات ولكن هناك بعض الأمور لم نتحدث عنها
كيف نضع صورة السحب؟
بإستخدامك الدالة setPixmap المتوفرة في الصف Qdrag ولإرجاعها استدعي الدالة pixmap .
كيف نحدد موقع مؤشر الفأرة في الصورة؟
setHotSpot وتستقبل في وسيطها الوحيد نقطة تحدد موقع الفأرة من الصورة ولكي تسترجعها فأنت بحاجة الى مايرادفها hotSpot .
كيف تعرف نافذة الأصل للعنصر والنافذة التي عليها العنصر ؟
الدالة source وهي التي تعيد لك مؤشرا من النوع Qwidget والتي تعيد النافذة التي تم سحب العنصر منها وفي المقابل الدال target والتي تعيد العنصر الذي يعيد مؤشر Qwidget الذي تم الإفلات بداخله او المرور من فوقه , جميع الدوال المذكورة هي من الصف QDrag .
في الحقيقة ان جميع ماتعلمته في السحب لا يفيدك في شيئ مالم تقم بتنفيذ الخطوة التالية وهي عملية البدء في عمليات السحب والإفلات بإستخدام الدالة exec والتي بوسيطها الأول والوحيد الذي يحدد طريقة نقل البيانات بواسطة الثابت التعدادي Qt::DropAction .

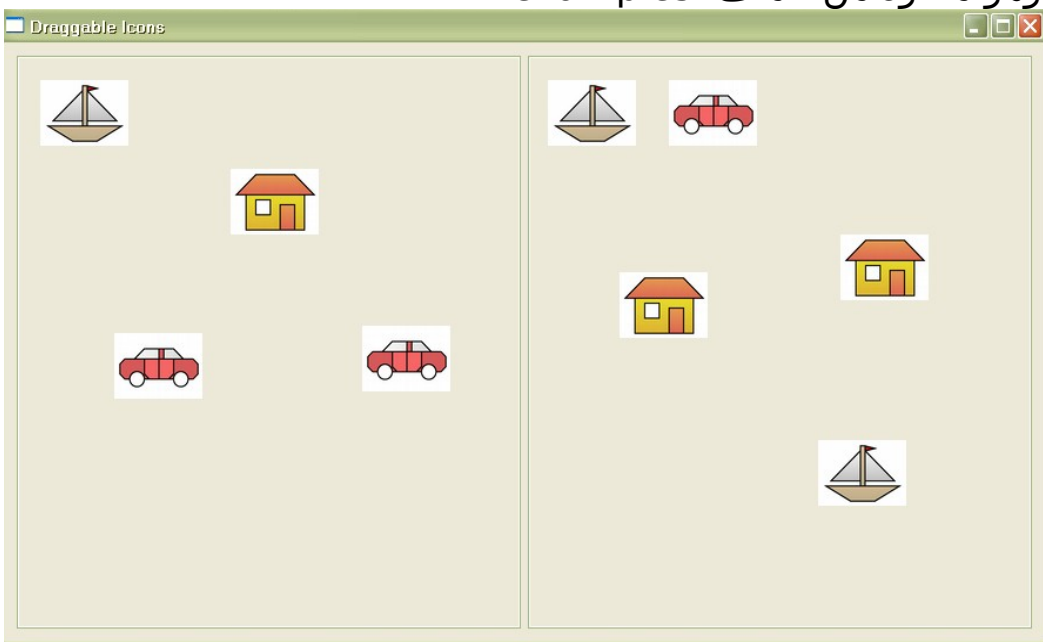
احداث السحب والإفلات

جميع صفوف الأحداث السابقة ترث من الصف QdropEvent ماعذا QdragLeaveEvent)) فلذلك لو قمنا بشراحة هذا الصف فإنه قد تعرفت عليهم جميعا .

لتحديد موقع الإفلات يمكنك إستخادام الدالة pos وهي تعيد قيم من النوع Qpoint وفي حال أردت ان تتعرف على البيانات التي تم سحبها يمكنك استخدام الأمر mimeData وهي يعيد مؤشر للنوع QmimeData ويمكنك استخدام الدالة source لمعرفة النافذة التي بدأت منها عملية السحب .
ولتحديد طريقة التعامل مع البيانات في عملية الإفلات يمكنك استخدام الدالة setDropAction وهي تستقبل وسيط وحيد من نوع ثابت تعدادي Qt::DropAction ويمكنك ان تتبعها بالدالة accept() لكي يتم تلقي الحدث ولا يتم البحث عن غيره او يمكنك استخدام الدالة `acceptProposedAction` ليرسل الموافقة على أي طريقة تعامل بيانات متوفرة في `exec` .

مثال:

في الصورة رقم 1 (مثال مصور) قمنا بتصوير برنامج بسيط لنقل البيانات بإستخدام البحث فلنحاول الآن تطبيق مثال شبيه لما يظهر في الرسم وهو مأخوذ من الملف examples .



شفرة الرأس

```
1. class DragWidget : public Qframe
2. {
3. public:
4. DragWidget(QWidget *parent=0);
5. protected:
6. void dragEnterEvent(QDragEnterEvent *event);
7. void dragMoveEvent(QDragMoveEvent *event);
8. void dropEvent(QDropEvent *event);
9. void mousePressEvent(QMouseEvent *event);
10.};
```

10-1هذا الصف يرث من Qframe لكي يتيح لنا تأطير النوافذ ولكن الأهم هنا ان جميع الدوال المحمية هي عبارة عن احداث اعيد تعريفها من الصف Qwidget فهي تحدث بشكل تلقائي عند الحدث وتعتمد على صفوف الأحداث في وسائطها .

```
1. #include "dragwidget.h"
2. DragWidget::DragWidget(QWidget *parent)
3. : QFrame(parent)
4. {
5. setMinimumSize(200, 200);
6. setFrameStyle(QFrame::Sunken | QFrame::StyledPanel);
7. setAcceptDrops(true);
8. QLabel *boatIcon = new QLabel(this);
9. boatIcon->setPixmap(QPixmap(":/images/boat.png"));
10.boatIcon->move(20, 20);
11.boatIcon->show();
12.boatIcon->setAttribute(Qt::WA_DeleteOnClose);
13.QLabel *carIcon = new QLabel(this);
14.carIcon->setPixmap(QPixmap(":/images/car.png"));
15.carIcon->move(120, 20);
16.carIcon->show();
17.carIcon->setAttribute(Qt::WA_DeleteOnClose);
18.QLabel *houseIcon = new QLabel(this);
19.houseIcon->setPixmap(QPixmap(":/images/house.png"));
20.houseIcon->move(20, 120);
21.houseIcon->show();
22.houseIcon->setAttribute(Qt::WA_DeleteOnClose);
23.}
```

1-7 قمنا في بداية الصف بتعيين اصغر ساحة للنافذة يمكن ان تحصل عليها ووضعنا شكل الإطار للنافذة ثم وجعلناه يقبل العناصر والبيانات التي تم فلتها عليه .

8-23 قمنا بوضع جميع العناصر(3) وهي عبارة عن label + صورة داخل النافذة وتحديد أماكنا وإعطائها ميزة الحذف عند الإقفال .

```
1. void DragWidget::mousePressEvent(QMouseEvent *event)
2. {
3. QLabel *child = static_cast<QLabel*>(childAt(event-
   >pos()));
4. if (!child)
5. return;
6. QPixmap pixmap = *child->pixmap();
7. QByteArray itemData;
8. QDataStream dataStream(&itemData,
   QIODevice::WriteOnly);
9. dataStream << pixmap << QPoint(event->pos() - child-
   >pos());
10.QMimeData *mimeData = new QMimeData;
11.mimeData->setData("application/x-dnditemdata",
   itemData);
```

```
12.QDrag *drag = new QDrag(this);
13.drag->setMimeData(mimeData);
14.drag->setPixmap(pixmap);
15.drag->setHotSpot(event->pos() - child->pos());
16.QPixmap tempPixmap = pixmap;
17.QPainter painter;
18.painter.begin(&tempPixmap);
19.painter.fillRect(pixmap.rect(), QColor(127, 127,
   127, 127));
20.painter.end();
21.child->setPixmap(tempPixmap);
22.if (drag->exec(Qt::CopyAction | Qt::MoveAction,
   Qt::CopyAction) == Qt::MoveAction)
23.child->close();
24.else {
25.child->show();
26.child->setPixmap(pixmap);
27.}
28.}
```

1-11 واذا افترضنا بدء دورة السحب فإنها سوف تبدأ بهذه الدالة وهي التي تحدث عند الضغط على أي زر من الفأرة بدون تحديد لتقوم هذه الدالة بالتعرف على ماهو هذا الشيئ الذي تم الضغط عليه ان كان label وهو ماسوف يكون بعد نجاح عملية القصر فإنه يقوم بأخذ الصورة من ال label ووضعناها داخل متغير QbyteArray بواسطة التيارات بالإضافة الى النقطة التي تم سحب العنصر منها ليقوم بإرسال البيانت اتي تم تجميعها في QbyteArray الى mimeData بمفتاح "application/x-dnditemdata", وهذا يعني ان هذه البيانات هي التي سوف يتم ارسالها اثنا عملية السحب .

12-15 قمنا بإنشاء المتغير drag من النوع Qdrag وقمن بوضع البيانات التي سوف يتم ارسالها بداخله وهي mimeData وثم وضعنا الصورة المأخوذة من ال label التي يتم سحبها لتكون واجهة للبيانات لا أكثر ومن ثم حددنا النقطة التي تحدد موقع الصورة الى الفأرة اثناء السحب لنجعلها نفس موقع الضغط عليها.

16-21 في كثير من المرات تجد اثناء عملية السحب يتم وضع طبقة شفافة اللون فوق العنصر الذي تم السحب منه وهذا ماقمنا به في هذه الأسطر فقمنا بأخذ مؤشر من الصورة التي على ال label ثم قمنا برسم مربع شفاف فوقه بمقدار النصف بإستخدام الصف Qpainter ولننهي الرسم ونعيد رفع الصورة الى ال label مرة أخرى .

22-28 هنا بدأنا بعملية السحب بإستخدام الدالة exec لتبقي عملية السحب مستمرة وفي حال تم الإفلات يرجع طريقة التي تم التعامل مع البيانات فيها فإن كانت طريقة التعامل نقل بيانات فإننا سوف نقوم بإغلاق ال label الذي تم السحب منه والا (نسخ بيانات) فإننا نقوم بجعلها مرئية ونضع الصورة بشكلها الطبيعة خالية من الطبقة الشفافة فوقها .

```
1. void DragWidget::dragMoveEvent(QDragMoveEvent
   *event)
2. {
3. if (event->mimeData()->hasFormat("application/x-
   dnditemdata")) {
4. if (event->source() == this) {
5. event->setDropAction(Qt::MoveAction);
6. event->accept();
7. } else {
8. event->acceptProposedAction();
9. }
10.} else {
11.event->ignore();
12.}
13.}
```

13-1 وهذه الدالة تحدث بشكل تلقائي اثناء عملية السحب وتحريك الفأرة بشكل آني ,ثم قمنا بإختبار ما اذا توفرت بيانات في المفتاح المعطى فإن نجحت قمنا بإختبار ما اذا النافذة المصدر لهذا العنصر أي التي تم سحب منها هذا العنصر هي نفسها هذه النافذة فإنه يعين حالة الإفلات على النقل ويرسل الموافقة اما ان لم تكن النافذة الحالية هي النافذة المصدر لهذا العنصر وإلا فإنه يتم ارسال الموافقة على نسخ البيانات وفي حال عدم توفر بيانات يتم الغاء عملية تحديد طريقة التعامل مع البيانات ويتجاهل الحدث .

```
1. void DragWidget::dragEnterEvent(QDragEnterEvent
   *event)
2. {
3. if (event->mimeData()->hasFormat("application/x-
   dnditemdata")) {
4. if (event->source() == this) {
5. event->setDropAction(Qt::MoveAction);
6. event->accept();
7. } else {
8. event->acceptProposedAction();
9. }
10.} else {
11.event->ignore();
12.}
13.}
```

وهذه الدالة تحدث بشكل تلقائي اثناء الدخول الى نافذة جديدة وهي مشابهة للدالة dragMoveEvent فلن تحتاج الى شرحة مرة أخرى .

```
1. void DragWidget::dropEvent(QDropEvent *event)
2. {
3. if (event->mimeData()->hasFormat("application/x-
   dnditemdata")) {
4. QByteArray itemData = event->mimeData()-
   >data("application/x-dnditemdata");
```

```
5. QDataStream dataStream(&itemData,
   QIODevice::ReadOnly);
6. QPixmap pixmap;
7. QPoint offset;
8. dataStream >> pixmap >> offset;
9. QLabel *newIcon = new QLabel(this);
10.newIcon->setPixmap(pixmap);
11.newIcon->move(event->pos() - offset);
12.newIcon->show();
13.newIcon->setAttribute(Qt::WA_DeleteOnClose);
14.if (event->source() == this) {
15.event->setDropAction(Qt::MoveAction);
16.event->accept();
17.} else {
18.event->acceptProposedAction();
19.}
20.}
21.else {
22. event->ignore();
23.}
24.}
```

1-13 وتحدث هذه الدالة بشكل تلقائي عن افلاتك للعنصر ثم أختبرنا في حال توفر بيانات من النوع application/x-dnditemdata فإنه يقوم بقراءتها بواسطة التيارات الثنائية وقد علمت سابقا اننا قمنا بتخزين صورة + موقع الفأرة من الصورة وهذا ماقمنا بقراءته ومن ثم صرحنا عن label جديد لنضع بداخله الصورة ونحرك ال label الى موضع الفأرة للنافذة ناقصا منه موضع الفأرة للصورة لتحديد الموقع بشكل أدق وثم قمنا بإضهارها ووضعنا له خاصية في حال قمنا بإقفال ال label فإن عملية حذفه تتم بسكل تلقائي.

14-24 وهي مشابهة للأسطر من 4-13 في الدالة السابقة .

التعامل مع clipboard

قامت المكتبة Qt بتوفير الصف Qclipboard ليتعامل مع ال clipboard مع مختلف أنظمة التشغيل .

ومن المعروف لك مسبقا ان ال clipBoard تعامل معه في الغالب في عمليات النسخ واللصق ولكن كمفهوم عام له فهو يتعامل مع نقل البيانات بين تطبيقات النظام ولكن هذه المرة سوف نتعامل معه بالسحب والإفلات لنخرج بذلك عن النمطية المستخدمة فيها .

كيف نرسل ونستقبل منه :

الأمر جدا بسيط كما قمنا بإستخدام mime data في نقل البيانات عبر السحب والإفلات داخل تطبيقك فإننا نستطيع ايضا استخدامه مع clipboard لكي لا يقون داخل التطبيق فقط انما يشمل النظام جميعه .

أستقبال clipboard من النظام :

يمكنك الحصول على مؤشر للصف Qclipboard يتعامل مع clipboard النظام مباشرة عبر الدالة

`QApplication::clipboard()` ليمكنك ارسال بيانات له عن طريق الدالة التالة setMimeData() من الصف Qclipboard او استقبال البيانات عن طريق الأمر mimeData() من نفس الصف ايضا .

مثال:

فكرة البرنامج انه عندما تقوم بسحب أي نص الى النافذة يتم اضافته مباشرة ال clipboard.

ونبدأ مع رأس الصف:

```
1. class D2cLabel : public QLabel
2. {
3. Q_OBJECT
4. public:
5. D2cLabel(QWidget *parent = 0);
6. void mousePressEvent(QMouseEvent *event);
7. void dragEnterEvent(QDragEnterEvent *event);
8. void dropEvent(QDropEvent *event);
9. protected:
10.QMimeData* cloneMimeData(const QMimeData *data);
11.}
```

هذا الصف يرث من Qlabel وعلى الأوامر السحب والإفلات وضغط أحد أزرار الفأرة والأمر cloneMimeData والذي سوفف يعيد بيانات منسوخة من mime data .

```
1. D2cLabel::D2cLabel(QWidget *parent)
2. : QLabel(parent)
3. {
4. setWordWrap(true);
5. setText(tr("<center>Drag from here to retrieve the text currently "
6. "located in the clipboard or fill the clipboard by "
7. "dragging text from abitrary places and dropping it here."
8. "</center"));
9. setAcceptDrops(true);
10.}
```

قمنا بكتابة النص على Qlabel بعدها جعلناه قابلا للإنتقال الى سطر جديد في حال أمتلئ السطر الحالي ثم جعلناه يتقبل البيانات التتي يتم اسقاطها عليه .

```
1. void D2cLabel::mousePressEvent(QMouseEvent *event)
2. {
3. if (event->button() == Qt::LeftButton) {
```

```
4. const QMimeData *mimeData = QApplication::clipboard()->mimeData();
5. if (!mimeData) return;
6. QDrag *drag = new QDrag(this);
7. drag->setMimeData(cloneMimeData(
8. QApplication::clipboard()->mimeData()));
9. drag->start();
10. }
11. }
```

عند الضغط على النافذة بالزر الأيسر من الفأرة فإنه يتم استقبال البيانات من clipboard النظام ليتم تخزينها في المؤشر mimeData وفي حال تلقيه أي بيانات يكمل ويصرح عن المؤشر drag من النوع Qdrag وادخلنا البيانت من clipboard اليه بعدما صنعنا نسخة لها بإستخدام الدالة cloneMimeData() التي قمنا بصناعتها وسوف نتكلم عنها لاحقا وبجأنا عملية السحب.

```
1. QMimeData* D2cLabel::cloneMimeData(const QMimeData *data)
2. {
3. if (!data)
4. return 0;
5. QMimeData *newData = new QMimeData;
6. foreach(QString format, data->formats())
7. newData->setData(format, data->data(format));
8. return newData;
9. }
```

وتقوم هذه الدالة كما ذكرنا مسبقا بنسخ بيانات البيانات وقد استخدمناها في الدالة السابقة لنسخ بيانات clipboard في السطر الخامس قمنا بالتصريح عن المتغير newData لنقوم بعملية التكرار للحاويات foreach وقمنا بسحب النوع لنضيفها الى newData في الوسيط الأول وفي الوسيط الثاني ادخلنا بيانات المتوفرة للنوع وسوف تتكرر هذه العملية الى ان تنتهي البيانات لنقوم في النهاية بإرجاع الدالة newData .

```
1. void D2cLabel::dragEnterEvent(QDragEnterEvent *event)
2. {
3. event->acceptProposedAction();
4. }
```

في حال تم الدخول الى النافذة فإنه يتم ارسال موافقة لطريقة نقل البيانا المقترحة .

```
1. void D2cLabel::dropEvent(QDropEvent *event)
2. {
3. if(event && event->mimeData()) {
4. QApplication::clipboard()->setMimeData(cloneMimeData
5. (event->mimeData()));
6. }
7. }
```

اذا ما حدث افلات للعنصر فإنك سوف تجد ان هذه الدالة تعمل بشكل تلقائي , واذا ماتوفرت بيانات فإنه يضيفها الى clipboard بواسطة الدالة setMimeData .

ملاحظة: في هذا البرنامج استخدمنا الدالة start لبدء عملية السحب وهو ماتم ازالته ابتداء من الإصدار 4.3 من قاموس Qt ولكن يمكنك استخدامه الى الآن .

اخوكم : محمد العبدلي