Qt drag and drop

QListWidget::setDragDropMode(QAbstractItemView::InternalMove) allows drag and drop. but nothing listens to it. and QListWidget has no appropriate signals.

https://forum.qt.io/topic/56326/qlistwidget-drag-and-drop

SOLVED QListWidget: DragDrop event Signal

> *I would like to know how to get a Qt signal when this internal dragdrop event occurs.*
>
> *it doesn't exist. What would you like to do on drop ? By the way, did you already read the drag and drop chapter for item views in Qt's documentation ?*

…

> *If the parent model index is valid, the drop occurred on an item.*

Model/View Programming#Using Drag and Drop with Item Views

qt is complicated

UX for move vs. swap?

apparently qt lets you grab a single item, then drop it either on an item or between items

but when every item is like 12 px tall, is it too hard to drop it in the right spot?

is it bad to allow both at the same time but one is swap and one is shift, and not separate them using a modifier key?

how does QListWidget/QAbstractItemModel handle moves?

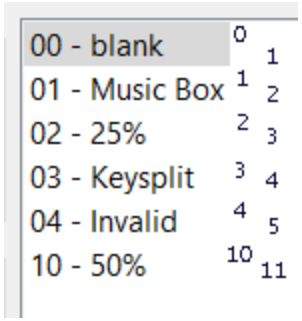
- auto model = _list->model();
- #define C(METHOD) \
    - connect(model, &QAbstractItemModel::METHOD, this, []() { \
        - qDebug() << #METHOD; \
    - })

- C(dataChanged);
- C(layoutAboutToBeChanged);
- C(layoutChanged);
- C(modelAboutToBeReset);
- C(modelReset);
- C(rowsAboutToBeInserted);
- C(rowsAboutToBeMoved);
- C(rowsAboutToBeRemoved);
- C(rowsInserted);
- C(rowsMoved);
- C(rowsRemoved);

dragging one item from one position to another (not swapping) produces:

- rowsAboutToBeMoved
- rowsMoved
    - parent=QModelIndex(-1,-1,0x0,QObject(0x0))
    - start=0
    - end=0
    - destination=QModelIndex(-1,-1,0x0,QObject(0x0))
    - row=2
- …
- rowsMoved
    - parent=QModelIndex(-1,-1,0x0,QObject(0x0))
    - start=16
    - end=16
    - destination=QModelIndex(-1,-1,0x0,QObject(0x0))
    - row=4

the position you drop an item determines the destination row. Note that when two adjacent items have hidden items in between, dragging to the top half of one item, and the bottom half of another, produce different results.



stack trace? https://gist.github.com/nyanpasu64/9d3e0ff17b7b5047b44dc90fe58c7249

- 1 gui::instrument_list::`anonymous namespace'::InstrumentListImpl::InstrumentListImpl::<unnamed-tag>::operator() instrument_list.cpp 88 0x7ff6756ee2bf
- …
- 8 QAbstractItemModel::rowsMoved moc_qabstractitemmodel.cpp 663 0x7ff81abccef5
- 9 QAbstractItemModel::endMoveRows qabstractitemmodel.cpp 3016 0x7ff81abce33c
- **10 QListModel::moveRows qlistwidget.cpp 324 0x7ff81d230593**
    - note that QListModel is private to QListWidget.
- **11 QAbstractItemModel::moveRow qabstractitemmodel.h 378 0x7ff81abe2297**
- **12 QListView::dropEvent qlistview.cpp 945 0x7ff81d20f092**
- **13 QListWidget::dropEvent qlistwidget.cpp 1979 0x7ff81d22e48d**

can QListView drag-and-drop produce edit operations with a parent?

does QAIV or QLV call QAIM::moveRows() with a parent? UPDATE: no.

QListView::move() calls QAIM::begin/endMoveRows with a null parent. idk if drag-drop calls it though.

control flow of move vs overwrite

- QAbstractItemView::startDrag(d->model->supportedDragActions());
    - what is QAbstractItemModel::supportedDragActions()?
        - QListModel does not override

                - apparently it's copy? idk.
        - QDrag::exec()
            - OLE bullshit
                - QListView::dropEvent()
                    - QAbstractItemViewPrivate::dropOn()
                        - **Compute the target of a drop.**
                        - if dropping between items, set row/col to valid. if dropping onto an item, set index to valid.
                    - if you drop between items, for each selected item, call QAIM::moveRow() and set dropEventMoved.
                        - normally dropEvent handles creating an item and startDrag handles removing what you dragged from. but in this case, QAIM::moveRow() removes the source, and dropEventMoved tells startDrag to not bother.
                        - *(If the first item fails to move by QAIM::moveRow(), all items are added/removed rather than moved. If the first item is moved by QAIM::moveRow() but the rest fail to move, the rest remain in place.)*
                        - See https://invent.kde.org/qt/qt/qtbase/-/commit/e356239397def8540ca2ec82274830a6c980a1a7 for how the system was built.
                        - **Is the selection moved with the items?**
                            - it's cleared by begin/endResetModel().
                            - the selection moves when you call QAIM::QAIM::changePersistentIndex[List]().
                    - otherwise...
                - become QAIV::dropEvent()
                    - if QAbstractItemViewPrivate::dropOn() again
                        - QAbstractListModel::dropMimeData(...row, col, index) inserts a new item. startDrag() removes the source of the item.
        - if move not copy, and !dropEventMoved, remove

why the fuck is "remove" so much shallower in the stack than "drop"?

Dragging a QTableView?

- QAbstractItemView::dropEvent
    - if (d->dropOn(event, &row, &col, &index)) {
        - const Qt::DropAction action = dragDropMode() == InternalMove ? Qt::MoveAction : event->dropAction();
        - **if (d->model->dropMimeData(event->mimeData(), action, row, col, index)) {**
            - *See also exotracker dropMimeData, QAIM dropMimeData, QALM/QATM.*
- clearOrRemove

If you choose to override QAIM::dropMimeData():

- When dragging onto an item (to replace it), index is valid and row is -1.
- When dragging between items (to move/insert between items), index is invalid and row is non-negative (within [0..rowCount()] inclusive).
- When dragging after the last item, index is invalid and row is -1.

If you want dragging an item after the last item to act like dragging to the bottom half of the last item, you must check if index is invalid, and if so, initiate a move/insert (if row == -1, use rowCount() instead). The original QAbstractListModel::dropMimeData() does this.

How does QListView handle drag and drop?

- QListView::dropEvent
    - QLVP::dropOn()
        - QAbstractItemViewPrivate::dropOn()
            - QLVP::position()
                - QAIVP::position()
        - ~~QListModeViewBase::dropOn()~~
        - if you drag and overwrite an item, dropOn returns true but writes -1 to row/col, but topIndex = what you dragged onto.
        - if you drag between items, dropOn returns true, writes row=... and col=0, but topIndex = invalid.
    - if (!topIndexDropped && !topIndex.isValid()) {
        - *wtf*
        - QPersistentModelIndex dropRow = model()->index(row, col, topIndex);
        - …
            - moveRow(QModelIndex(), pIndex.row(), QModelIndex(), r)

seriously why the fuck does QListView just discard

---

why does Qt special-case left-to-right list views? idk. (it doesn't matter, we use top-to-bottom.)

how does Qt determine what drop actions are valid?

- QAbstractItemViewPrivate::position(const QPoint &pos, const QRect &rect, const QModelIndex &index) const
    - QAbstractItemView::DropIndicatorPosition r = QAbstractItemView::OnViewport;
    - if (!overwrite) {
        - const int margin = qBound(2, qRound(qreal(rect.height()) / 5.5), 12);
        - if (pos.y() - rect.top() < margin) {
            - r = QAbstractItemView::AboveItem;
        - } else if (rect.bottom() - pos.y() < margin) {
            - r = QAbstractItemView::BelowItem;
        - } else if (rect.contains(pos, true)) {
            - r = QAbstractItemView::OnItem;
    - } else {
        - QRect touchingRect = rect;
        - touchingRect.adjust(-1, -1, 1, 1);
        - if (touchingRect.contains(pos, false)) {
            - r = QAbstractItemView::OnItem;
    - if (r == QAbstractItemView::OnItem && (!(model->flags(index) & Qt::ItemIsDropEnabled)))
        - r = pos.y() < rect.center().y() ? QAbstractItemView::AboveItem : QAbstractItemView::BelowItem;
    - return r;

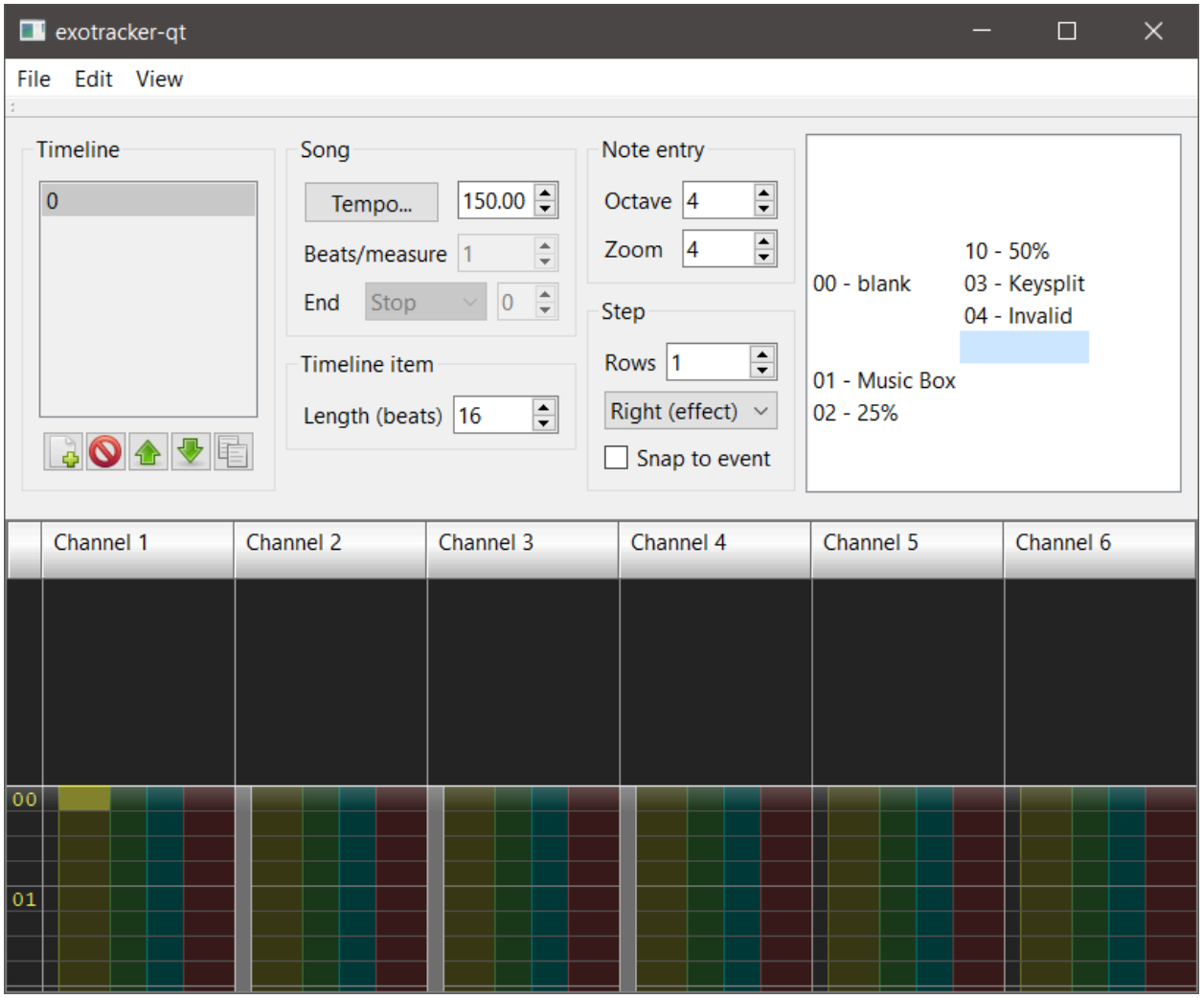
what's QAbstractItemViewPrivate::overwrite? (set by QAbstractItemView::setDragDropOverwriteMode().)

what's QAIM::flags() & Qt::ItemIsDropEnabled? (whether a model allows dropping an item *onto* an item)

bug

- _list->setSelectionMode(QAbstractItemView::SingleSelection);

- _list->setDragEnabled(true);
- _list->viewport()->setAcceptDrops(true);
- _list->setDropIndicatorShown(true);
- _list->setDefaultDropAction(Qt::MoveAction);
- _list->setDragDropMode(QAbstractItemView::InternalMove);
- _list->setDragDropOverwriteMode(true);



https://bugreports.qt.io/browse/QTBUG-96248

qt5.15=qt6

custom-defined drag-drop behavior?

maybe write a QListView subclass with a custom dropEvent() that swaps if no button held, and moves if Ctrl held? (or someday copies with Shift held?)

and change the cursor icon when holding Ctrl, and write a status bar message saying "hold Ctrl to move"?

seeing how QAbstractItemViewPrivate::position() is implemented, I should leave QAIM::dragDropOverwriteMode to true, and toggle QAIM::flags().ItemIsDropEnabled based on whether in swap or move mode… but is it possible to convince QListWidget to swap rather than overwrite?

QListWidget::flags() returns ItemIsSelectable|ItemIsDragEnabled|ItemIsUserCheckable|ItemIsEnabled.

how does QListWidget hide items?

How does QListWidget hide items? Does the internal model have a custom role for "is hidden" which QListView understands?

https://doc.qt.io/qt-5/qlistview.html#setRowHidden suggests it's purely a view property, not part of the model.

QListWidgetItem::isHidden() delegates to QListWidget.

how does QAbstractItemModel handle swaps?

idk. QAbstractItemModel has no provision for swaps. though you could use "move as a child" as a synonym for swap.

how does bamboo handle swaps?

it has a custom class, DropDetectListWidget, that emits a custom signal itemDroppedAtItemIndex() upon drag-and-drop.

if the signal is not emitted, drag-and-drop neither mutates the document nor reorders the widget.

how does Ui_MainWindow::instrument allow drag and drop, but ignore it actually occurring?

- ui->instrumentList->installEventFilter(this);
    - if (watched == ui->instrumentList) {
        - if (event->type() == QEvent::FocusIn) updateMenuByInstrumentList();

        - return false;
- QObject::connect(ui->instrumentList, &DropDetectListWidget::itemDroppedAtItemIndex,
    - this, &MainWindow::swapInstruments);
- <widget class="DropDetectListWidget" name="instrumentList">
    - contextMenuPolicy=Qt::CustomContextMenu
    - showDropIndicator=false
        - true will mistakenly show a line when dropping.
    - dragEnabled=true
    - dragDropMode=QAbstractItemView::DragDrop
    - defaultDropAction=Qt::ActionMask
        - 0xff
        - switching to MoveAction causes drag-and-drop to reorder instruments without informing the document.

Advise on requirements for version control of Qt UI XML files .ui is hell.

QAIM is hell

https://bugreports.qt.io/browse/QTBUG-94226

> *Ah, doh. Yeah, ignoring that event if the data in the model isn't moved is not helpful, since the icon-mode listview doesn't move data in the model, but only rearranges the layout. This conflating of list and icon view into a single class is such a terrible, terrible design…*

abandoning QAIM/V

why is qabstractitemview so "enterprisey"

simultaneously too open-ended (go subclass Q[Foo]ItemModel, figure the roles out yourself through trial and error?)

not flexible enough (QListView imposes its view of "drag-and-drop is either a move or an overwrite" and takes responsibility for dictating which item is moved where, by calling QAbstractItemModel::moveRow()... it's hard to instead have it expose "user dragged item 1 onto item 2" and "user dragged item 1 between items 2 and 3" to the model, and the model chooses how to react, chooses how to update non-Qt data stores)

and too buggy

---

i think i'm going to copy bambootracker. use QListWidget for cosmetics, but hook it to output events used to edit the document and reload the widget.

subclass QListWidget and add custom modifier callbacks to toggle between swap and move, enabling drop hints iff in Move mode.

bambootracker copy erases items when dragged

https://gist.github.com/nyanpasu64/9760afbdc6fbdac413f11a7284450416

- 8 QAbstractItemModel::rowsRemoved moc_qabstractitemmodel.cpp 606 0x7ffff6c10533
- 9 QAbstractItemModel::endRemoveRows qabstractitemmodel.cpp 2835 0x7ffff6c18e75
- 10 QListModel::removeRows qlistwidget.cpp 292 0x7ffff7d32f49
- 11 QAbstractItemViewPrivate::clearOrRemove qabstractitemview.cpp 4322 0x7ffff7ced23c
- 12 QAbstractItemView::startDrag qabstractitemview.cpp 3717 0x7ffff7cef509
    - QModelIndexList indexes = d->selectedDraggableIndexes();
    - if (indexes.count() > 0) {
        - QMimeData *data = d->model->mimeData(indexes);
        - if (!data)
            - return;
        - QRect rect;
        - QPixmap pixmap = d->renderToPixmap(indexes, &rect);
        - rect.adjust(horizontalOffset(), verticalOffset(), 0, 0);
        - QDrag *drag = new QDrag(this);
        - drag->setPixmap(pixmap);
        - drag->setMimeData(data);
        - drag->setHotSpot(d->pressedPosition - rect.topLeft());
        - Qt::DropAction defaultDropAction = Qt::IgnoreAction;
        - if (dragDropMode() == InternalMove)
            - supportedActions &= ~Qt::CopyAction;
        - if (d->defaultDropAction != Qt::IgnoreAction && (supportedActions & d->defaultDropAction))

        - defaultDropAction = d->defaultDropAction;
    - else if (supportedActions & Qt::CopyAction && dragDropMode() != QAbstractItemView::InternalMove)
        - defaultDropAction = Qt::CopyAction;
    - d->dropEventMoved = false;
    - if (drag->exec(supportedActions, defaultDropAction) == Qt::MoveAction && !d->dropEventMoved)
        - d->clearOrRemove();
    - d->dropEventMoved = false;
    - // Reset the drop indicator
    - d->dropIndicatorRect = QRect();
    - d->dropIndicatorPosition = OnItem;
- 13 QAbstractItemView::mouseMoveEvent qabstractitemview.cpp 1834 0x7ffff7cedc29
- 14 QListView::mouseMoveEvent qlistview.cpp 783 0x7ffff7d2c7f8
- 15 QListView::mouseMoveEvent qlistview.cpp 778 0x7ffff7d2c7f8
- 16 QWidget::event qwidget.cpp 9020 0x7ffff7aa0fce
- 17 QFrame::event qframe.cpp 550 0x7ffff7b51983

fix

setDefaultDropAction(Qt::ActionMask)...

Qt::IgnoreAction/CopyAction fails.

i think qt isn't designed to handle ActionMask, this is driving Qt into an unexpected state, and it's a coincidence it works.

and weirdly enough, the drag cursor is "copy" (green +) on KDE, for both Bamboo and Exo.

this is a hack.

[done] making custom QAIM work

```
Qt::DropActions supportedDragActions() const override {
    return Qt::MoveAction;
}
```

```
Qt::DropActions supportedDropActions() const override {
    return Qt::MoveAction;
}

Qt::ItemFlags flags(const QModelIndex &index) const override {
    Qt::ItemFlags defaultFlags = QAbstractListModel::flags(index);

    if (index.isValid())
        return Qt::ItemIsDragEnabled | Qt::ItemIsDropEnabled | defaultFlags;
    else
        return Qt::ItemIsDropEnabled | defaultFlags;
}
```

and now drag works… and does nothing.

fun fact: QSortFilterProxyModel doesn't know how to reorder rows, because there's holes.

using QSortFilterProxyModel rather than display-level hiding means you can't distinguish between "drag to top edge of one item" and "drag to bottom edge of the item above".

TODO switch to view-level item hiding

---

and now, dragging an item allows me to drag onto an item, onto its top edge, or onto its bottom edge. why? See How does QListView handle drag and drop?.

- how do i make it "drag onto" by default?
    - QListView::setDragDropOverwriteMode(true)
- and later on, how do i make it "move to gap" with a modifier held, as indicated in the status bar?
    - by having InstrumentListModel::flags() disable ItemIsDropEnabled on command.

---

i managed to make QListView expose an appropriate front-end for swapping *and* rotating. it is the job of InstrumentListModel to override the virtual methods, and such.

…

anyway i finally found how to nudge qt's model-view system into getting out of the way, and have the View act as an input for the user to symbolically operate on a model, which can interpret the user's actions however i want

in my current understanding, QAbstractItemModel's virtual methods are commands, and the ~~signals~~ EDIT: methods emitting signals are actions it takes to respond to commands.

so moveRows() doesn't actually have to emit rowsAboutToBeMoved followed by emit rowsMoved, you can respond to the command however you want

(of course, i'm sure that one false step and the complexity returns... like just look at the mess that is that one QAbstractListView bug fix)

QListView discards overwrite drags

PROBLEM: when i drag one item *onto* another item in a QListView, the model is never informed!

How does QListView handle drag and drop?

oh, of course QListView silently discards all attempts to drag one item onto another with ItemIsDropEnabled set, instead of informing the model.

Who else calls QAbstractItemViewPrivate::dropOn() and actually uses "drag onto"?

https://stackoverflow.com/questions/21821106/stop-qlistview-from-deleting-entries-on-drag-drop

this person managed to get overwriting to work perfectly fine. how?

reinterpreting QAbstractItemModel::removeRows() to instead swap?

not really possible. removeRows() doesn't say what's being dragged onto it.

https://gist.github.com/nyanpasu64/ba245d8b33aa40a6a3f8b4103c64a456

see bambootracker copy erases items when dragged.

changing the model to interpret "drag onto" as "swap"

solution? perhaps override QAIM::dropEvent(), and bypass QListView::dropEvent() and call QAbstractItemView::dropEvent() directly? or don't bypass. either way doesn't matter.

in any case, QAbstractItemView::dropEvent() calls QAIM::dropMimeData(), which is responsible for (if row/col is set) inserting or (if index is set) replacing.

Solution: in InstrumentListModel, reimplement QAbstractListModel::dropMimeData() to swap. (what about QAbstractItemModel::canDropMimeData?)

- you don't know the source of the data
    - you do, QMimeData stores it.
- you can't prevent it from being deleted
    - solution: when the user move-rotates a model, use Qt::MoveAction. when the user swaps items, use Qt::CopyAction so the source isn't deleted.
        - but Qt::CopyAction is unsupported in QListView::InternalMove.
    - or we could use Qt::MoveAction and ignore all moves… evil.

and when InstrumentListModel tells InstrumentListModel to swap two items, it can use its own method (command) that isn't in the QAIM vocabulary, and nobody will know the difference.

you call the method, it creates a transaction, applies an EditBox, ~~and tells qt that "all data changed"...~~ creates a StateTransaction which tells qt that data changed.

- it's bad imo for ILM to suppress StateTransaction's reactivity system and emit its own fine-grained ILM signals.
- undo-generated StateTransaction lacks fine-grained data change info, so ILM won't expose it for edits it makes to itself either.
- anything other than "model reset" requires StateTransaction to emit a "before data changed" signal when it first lends document_mut. this sucks.

drag and drop

- [ ] override QAbstractItemView::startDrag() to create QDrop ourself?
- [ ] ~~fudge QDrag::exec()~~

    - not possible, both the return value and the type of drag depend on model.supportedDragActions and view.defaultdragaction.
- [x] ~~just eat all removeRows().~~

Dragging an item past the end

In "move to gap" mode (QAIM::flags() returns ItemIsDropEnabled only on invalid indices), when you drag an item past the last item visible, it invokes QAIM::moveRows(...destinationChild = rowCount()).

If I ever implement instrument reordering, this should be clamped to "last non-empty slot + 1" if empty slots are hidden, because nobody deliberately intends to move an instrument to after slot FF.