```
#ifndef QT_NO_PRINTER
#include <QtPrintSupport/QPrinter>
#endif
```

```
}


#include <QtPrintSupport/QPrintDialog>

#include "arrow.h"
#include "diagramitem.h"
#include "diagramscene.h"
#include "diagramtextitem.h"
#include "mainwindow.h"

#include <QtWidgets>

const int InsertTextButton = 10;

MainWindow::MainWindow() {
    createActions();
    createToolBox();
    createMenus();

    scene = new DiagramScene(itemMenu, this);
    scene->setSceneRect(QRectF(0,0,100,200));//fffff
    connect(scene, SIGNAL(itemInserted(DiagramItem*)),
            this, SLOT(itemInserted(DiagramItem*)));
    connect(scene, SIGNAL(textInserted(QGraphicsTextItem*)),
            this, SLOT(textInserted(QGraphicsTextItem*)));
    connect(scene, SIGNAL(arrowInserted()),
            this, SLOT(backupUndostack()));
    connect(scene, SIGNAL(textChanged()),
            this, SLOT(backupUndostack()));
    connect(scene, SIGNAL(itemSelected(QGraphicsItem*)),
            this, SLOT(itemSelected(QGraphicsItem*)));
    connect(scene, SIGNAL(scaleChanging(int)),
            this, SLOT(sceneScaleZooming(int)));

    createToolbars();


    QHBoxLayout *layout = new QHBoxLayout;
    layout->addWidget(toolBox);
    view = new DiagramView(scene);
    layout->addWidget(view);

    connect(view, SIGNAL(needsUndoBackup()), this, SLOT(backupUndostack()));

    QWidget *widget = new QWidget;
    widget->setLayout(layout);

    setCentralWidget(widget);
    setWindowTitle(tr("Diagramscene"));
    setUnifiedTitleAndToolBarOnMac(true);

    undoStack.backup(QList<QGraphicsItem*>());
}

void MainWindow::backgroundButtonGroupClicked(QAbstractButton *button)
```

```
{
    QList<QAbstractButton *> buttons = backgroundButtonGroup->buttons();
    foreach (QAbstractButton *myButton, buttons) {
        if (myButton != button)
            button->setChecked(false);
    }
    QString text = button->text();
    if (text == tr("Blue Grid"))
        scene->setBackgroundBrush(QPixmap(":/images/background1.png"));
    else if (text == tr("White Grid"))
        scene->setBackgroundBrush(QPixmap(":/images/background2.png"));
    else if (text == tr("Gray Grid"))
        scene->setBackgroundBrush(QPixmap(":/images/background3.png"));
    else
        scene->setBackgroundBrush(QPixmap(":/images/background4.png"));

    scene->update();
    view->update();
}

void MainWindow::buttonGroupClicked(int id) {
    QList<QAbstractButton *> buttons = buttonGroup->buttons();
    QAbstractButton* clickedButton = buttonGroup->button(id);

    // set other button unchecked
    foreach (QAbstractButton *button, buttons) {
        if (clickedButton != button)
            button->setChecked(false);
    }
    // simply set objButton unchecked if already checked
    if (!clickedButton->isChecked()) {
        scene->setMode(DiagramScene::Mode(pointerTypeGroup->checkedId()));
        return;
    }

    if (id == InsertTextButton) {
        scene->setMode(DiagramScene::InsertText);
    } else {
        scene->setItemType(DiagramItem::DiagramType(id));
        scene->setMode(DiagramScene::InsertItem);
    }
}

void MainWindow::copyItem() {
    foreach(QGraphicsItem* p, pasteBoard) {
        delete p;
    }
    pasteBoard = cloneItems(scene->selectedItems());
    qDebug() << pasteBoard.size();
}

void MainWindow::pasteItem()
{
    QList<QGraphicsItem*> pasteBoardCopy(cloneItems(pasteBoard));
    foreach(QGraphicsItem* p, scene->items()) p->setSelected(false);

    foreach(QGraphicsItem* item, pasteBoard)
    {
        if (item->type() != Arrow::Type)
        {
            item->setPos(item->scenePos() + QPointF(20, 20));
            item->setZValue(item->zValue() + 0.1);  // raise a little bit
        }
        scene->addItem(item);
        item->setSelected(true);
    }
    pasteBoard.swap(pasteBoardCopy);
    if (!scene->selectedItems().empty())
        undoStack.backup(cloneItems(scene->items()));
}

void MainWindow::cutItem() {
    copyItem();
    deleteItem();
}
void MainWindow::printScene()
{
    QPrinter printer;

    if (QPrintDialog(&printer).exec() == QDialog::Accepted) {
        QPainter painter(&printer);

        painter.setRenderHint(QPainter::Antialiasing);
        //QRectF* target= new QRectF();

        scene->render(&painter,QRectF(0,0,100,200));
    }
}

void MainWindow::deleteItem() {
    bool needsBackup = !scene->selectedItems().empty();
    scene->deleteItems(scene->selectedItems());
    if (needsBackup)
        undoStack.backup(cloneItems(scene->items()));
}

void MainWindow::undo() {
    if (undoStack.isEmpty()) return;
    // sweep away all items
    scene->deleteItems(scene->items());
    QList<QGraphicsItem*> undoneItems = cloneItems(undoStack.undo());
    foreach(QGraphicsItem* item, undoneItems) {
        qDebug() << item << "--------- add item";
        scene->addItem(item);
    }
    // update arrows
    foreach(QGraphicsItem* item, undoneItems) {
        if (item->type() == Arrow::Type)
            qgraphicsitem_cast<Arrow*>(item)->updatePosition();
    }
}

void MainWindow::redo() {
    if (undoStack.isFull()) return;
    scene->deleteItems(scene->items());
    QList<QGraphicsItem*> redoneItems = cloneItems(undoStack.redo());
    foreach(QGraphicsItem* item, redoneItems) {
        scene->addItem(item);
    }

    // update arrows
    foreach(QGraphicsItem* item, redoneItems) {
        if (item->type() == Arrow::Type)
            qgraphicsitem_cast<Arrow*>(item)->updatePosition();
    }
}

void MainWindow::groupItems() {
    QGraphicsItemGroup* group = scene->createItemGroup(scene->selectedItems());
    group->setFlag(QGraphicsItem::ItemIsMovable, true);
    group->setFlag(QGraphicsItem::ItemIsSelectable, true);
    group->setFlag(QGraphicsItem::ItemSendsGeometryChanges, true);
    scene->addItem(group);
    backupUndostack();
}

void MainWindow::ungroupItems() {
    foreach(QGraphicsItem* p, scene->selectedItems()) {
        if (p->type() == QGraphicsItemGroup::Type) {
            scene->destroyItemGroup(qgraphicsitem_cast<QGraphicsItemGroup*>(p));
        }
    }
    backupUndostack();
}

void MainWindow::pointerGroupClicked(int)
{
    // set all buttons in toolbox unchecked
    foreach(QAbstractButton* b, buttonGroup->buttons()) {
        b->setChecked(false);
    }
    scene->setMode(DiagramScene::Mode(pointerTypeGroup->checkedId()));
}

void MainWindow::bringToFront()
{
    if (scene->selectedItems().isEmpty())
        return;
    QGraphicsItem *selectedItem = scene->selectedItems().first();
    QList<QGraphicsItem *> overlapItems = selectedItem->collidingItems();

    qreal zValue = 0;
    foreach (QGraphicsItem *item, overlapItems) {
        if (item->zValue() >= zValue && item->type() == DiagramItem::Type)
            zValue = item->zValue() + 0.1;
    }
    selectedItem->setZValue(zValue);
    backupUndostack();
}
//! [5]

//! [6]
void MainWindow::sendToBack()
{
    if (scene->selectedItems().isEmpty())
        return;

    QGraphicsItem *selectedItem = scene->selectedItems().first();
    QList<QGraphicsItem *> overlapItems = selectedItem->collidingItems();

    qreal zValue = 0;
    foreach (QGraphicsItem *item, overlapItems) {
        if (item->zValue() <= zValue && item->type() == DiagramItem::Type)
            zValue = item->zValue() - 0.1;
    }
    selectedItem->setZValue(zValue);
    backupUndostack();
}
//! [6]

//! [7]
void MainWindow::itemInserted(DiagramItem *item)
{
    pointerTypeGroup->button(int(DiagramScene::MoveItem))->setChecked(true);
    scene->setMode(DiagramScene::Mode(pointerTypeGroup->checkedId()));
    buttonGroup->button(int(item->diagramType()))->setChecked(false);
    backupUndostack();
}
//! [7]

//! [8]
void MainWindow::textInserted(QGraphicsTextItem *)
{
    buttonGroup->button(InsertTextButton)->setChecked(false);
    scene->setMode(DiagramScene::Mode(pointerTypeGroup->checkedId()));
}

void MainWindow::backupUndostack() {
    undoStack.backup(cloneItems(scene->items()));
}
```

```
//! [8]

//! [9]
void MainWindow::currentFontChanged(const QFont &)
{
    handleFontChange();
}
//! [9]

//! [10]
void MainWindow::fontSizeChanged(const QString &)
{
    handleFontChange();
}
//! [10]

//! [11]
void MainWindow::sceneScaleChanged(const QString &scale)
{
    double newScale = scale.toDouble() / 100.0;
    QMatrix oldMatrix = view->matrix();
    view->resetMatrix();
    view->translate(oldMatrix.dx(), oldMatrix.dy());
    view->scale(newScale, newScale);
}

void MainWindow::sceneScaleZooming(int delta) {
    int changingPercent = delta > 0 ? 10 : -10;
    QString comboText = sceneScaleCombo->currentText();
    int newScale = comboText.toInt() + changingPercent;

    if (newScale > 0 && newScale <= 200) {
        sceneScaleCombo->setCurrentText(QString().number(newScale));
    }
}
//! [11]

//! [12]
void MainWindow::textColorChanged()
{
    textAction = qobject_cast<QAction *>(sender());
    fontColorToolButton->setIcon(createColorToolButtonIcon(
                                     ":/images/textpointer.png",
                                     qvariant_cast<QColor>(textAction->data())));
    textButtonTriggered();
    undoStack.backup(cloneItems(scene->items()));
}
//! [12]

//! [13]
void MainWindow::itemColorChanged()
{
    fillAction = qobject_cast<QAction *>(sender());
    fillColorToolButton->setIcon(createColorToolButtonIcon(
                                 ":/images/floodfill.png",
                                 qvariant_cast<QColor>(fillAction->data())));
    fillButtonTriggered();
    undoStack.backup(cloneItems(scene->items()));
}
//! [13]

//! [14]
void MainWindow::lineColorChanged()
{
    lineAction = qobject_cast<QAction *>(sender());
    lineColorToolButton->setIcon(createColorToolButtonIcon(
                                 ":/images/linecolor.png",
                                 qvariant_cast<QColor>(lineAction->data())));
    lineButtonTriggered();
}
//! [14]

//! [15]
void MainWindow::textButtonTriggered()
{
    scene->setTextColor(qvariant_cast<QColor>(textAction->data()));
}
//! [15]

//! [16]
void MainWindow::fillButtonTriggered()
{
    scene->setItemColor(qvariant_cast<QColor>(fillAction->data()));
}
//! [16]

//! [17]
void MainWindow::lineButtonTriggered()
{
    scene->setLineColor(qvariant_cast<QColor>(lineAction->data()));
}
//! [17]

//! [18]
void MainWindow::handleFontChange()
{
    QFont font = fontCombo->currentFont();
    font.setPointSize(fontSizeCombo->currentText().toInt());
    font.setWeight(boldAction->isChecked() ? QFont::Bold : QFont::Normal);
    font.setItalic(italicAction->isChecked());
    font.setUnderline(underlineAction->isChecked());

    scene->setFont(font);
}
//! [18]
```

```cpp
//! [19]
void MainWindow::itemSelected(QGraphicsItem *item)
{
    DiagramTextItem *textItem =
    qgraphicsitem_cast<DiagramTextItem *>(item);

    QFont font = textItem->font();
    //fontCombo->setCurrentFont(font);
    fontSizeCombo->setEditText(QString().setNum(font.pointSize()));
    boldAction->setChecked(font.weight() == QFont::Bold);
    italicAction->setChecked(font.italic());
    underlineAction->setChecked(font.underline());
}
//! [19]

//! [20]
void MainWindow::about()
{
    QMessageBox::about(this, tr("About Diagram Scene"),
                       tr("A drawing tool based on Qt Example."));
}

void MainWindow::createToolBox()
{
    buttonGroup = new QButtonGroup(this);
    buttonGroup->setExclusive(false);
    connect(buttonGroup, SIGNAL(buttonClicked(int)),
            this, SLOT(buttonGroupClicked(int)));
    QGridLayout *layout = new QGridLayout;
   // layout->addWidget(createCellWidget(tr("Conditional"),
DiagramItem::Conditional), 0, 0);

    layout->addWidget(createCellWidget(tr("Step"), DiagramItem::Step),0, 0);
    layout->addWidget(createCellWidget(tr("Step1"), DiagramItem::Step1),1, 0);
    layout->addWidget(createCellWidget(tr("Step2"), DiagramItem::Step2),2, 0);
    layout->addWidget(createCellWidget(tr("Step3"), DiagramItem::Step3),3, 0);
    layout->addWidget(createCellWidget(tr("Step4"), DiagramItem::Step4),4, 0);
    layout->addWidget(createCellWidget(tr("Step5"), DiagramItem::Step5),5, 0);
    layout->addWidget(createCellWidget(tr("Step6"), DiagramItem::Step6),6, 0);
    layout->addWidget(createCellWidget(tr("Step7"), DiagramItem::Step7),7, 0);
    layout->addWidget(createCellWidget(tr("Step8"), DiagramItem::Step8),8, 0);
    layout->addWidget(createCellWidget(tr("Step9"), DiagramItem::Step9),9, 0);
    layout->addWidget(createCellWidget(tr("Step10"), DiagramItem::Step10),10, 0);
    layout->addWidget(createCellWidget(tr("Step11"), DiagramItem::Step11),11, 0);
    layout->addWidget(createCellWidget(tr("Step12"), DiagramItem::Step12),12, 0);
    layout->addWidget(createCellWidget(tr("Step13"), DiagramItem::Step13),13, 0);
    layout->addWidget(createCellWidget(tr("Step14"), DiagramItem::Step14),14, 0);
    layout->addWidget(createCellWidget(tr("Step15"), DiagramItem::Step15),15, 0);
    layout->addWidget(createCellWidget(tr("Step16"), DiagramItem::Step16),16, 0);
    layout->addWidget(createCellWidget(tr("Step17"), DiagramItem::Step17),17, 0);
    layout->addWidget(createCellWidget(tr("Step18"), DiagramItem::Step18),18, 0);
    layout->addWidget(createCellWidget(tr("Step19"), DiagramItem::Step19),19, 0);
}



    QToolButton *textButton = new QToolButton;
    textButton->setCheckable(true);
    buttonGroup->addButton(textButton, InsertTextButton);
    textButton->setIcon(QIcon(QPixmap(":/images/textpointer.png")));
    textButton->setIconSize(QSize(50, 50));
    QGridLayout *textLayout = new QGridLayout;
    textLayout->addWidget(textButton, 0, 0, Qt::AlignHCenter);
    textLayout->addWidget(new QLabel(tr("Text")), 1, 0, Qt::AlignCenter);
    QWidget *textWidget = new QWidget;
    textWidget->setLayout(textLayout);
    //layout->addWidget(textWidget, 4, 0);

    layout->setRowStretch(3, 10);
    layout->setColumnStretch(2, 10);

    QWidget *itemWidget = new QWidget;
    itemWidget->setLayout(layout);

    backgroundButtonGroup = new QButtonGroup(this);
    connect(backgroundButtonGroup, SIGNAL(buttonClicked(QAbstractButton*)),
            this, SLOT(backgroundButtonGroupClicked(QAbstractButton*)));

    QGridLayout *backgroundLayout = new QGridLayout;
    backgroundLayout->addWidget(createBackgroundCellWidget(tr("Blue Grid"),
                                                           ":/images/
background1.png"), 0, 0);
    backgroundLayout->addWidget(createBackgroundCellWidget(tr("White Grid"),
                                                           ":/images/
background2.png"), 0, 1);
    backgroundLayout->addWidget(createBackgroundCellWidget(tr("Gray Grid"),
                                                           ":/images/
background3.png"), 1, 0);
    backgroundLayout->addWidget(createBackgroundCellWidget(tr("No Grid"),
                                                           ":/images/
background4.png"), 1, 1);

    backgroundLayout->setRowStretch(2, 10);
    backgroundLayout->setColumnStretch(2, 10);

    QWidget *backgroundWidget = new QWidget;
    backgroundWidget->setLayout(backgroundLayout);

    toolBox = new QToolBox;
    toolBox->setSizePolicy(QSizePolicy(QSizePolicy::Maximum,
QSizePolicy::Ignored));
    toolBox->setMinimumWidth(itemWidget->sizeHint().width());
    toolBox->addItem(itemWidget, tr("Basic Flowchart Shapes"));
    toolBox->addItem(backgroundWidget, tr("Backgrounds"));
}

void MainWindow::createActions()
```

```
{
    toFrontAction = new QAction(QIcon(":/images/bringtofront.png"),
                                tr("Bring to &Front"), this);
    toFrontAction->setShortcut(tr("Ctrl+F"));
    toFrontAction->setStatusTip(tr("Bring item to front"));
    connect(toFrontAction, SIGNAL(triggered()), this, SLOT(bringToFront()));

    sendBackAction = new QAction(QIcon(":/images/sendtoback.png"), tr("Send to
&Back"), this);
    sendBackAction->setShortcut(tr("Ctrl+T"));
    sendBackAction->setStatusTip(tr("Send item to back"));
    connect(sendBackAction, SIGNAL(triggered()), this, SLOT(sendToBack()));

    deleteAction = new QAction(QIcon(":/images/delete.png"), tr("&Delete"), this);
    deleteAction->setShortcut(tr("Delete"));
    deleteAction->setStatusTip(tr("Delete item from diagram"));
    connect(deleteAction, SIGNAL(triggered()), this, SLOT(deleteItem()));

    exitAction = new QAction(tr("E&xit"), this);
    exitAction->setShortcuts(QKeySequence::Quit);
    exitAction->setStatusTip(tr("Quit Scenediagram example"));
    connect(exitAction, SIGNAL(triggered()), this, SLOT(close()));

    boldAction = new QAction(tr("Bold"), this);
    boldAction->setCheckable(true);
    QPixmap pixmap(":/images/bold.png");
    boldAction->setIcon(QIcon(pixmap));
    boldAction->setShortcut(tr("Ctrl+B"));
    connect(boldAction, SIGNAL(triggered()), this, SLOT(handleFontChange()));

    italicAction = new QAction(QIcon(":/images/italic.png"), tr("Italic"), this);
    italicAction->setCheckable(true);
    italicAction->setShortcut(tr("Ctrl+I"));
    connect(italicAction, SIGNAL(triggered()), this, SLOT(handleFontChange()));

    printAction = new QAction(QIcon(":/images/italic.png"),tr("Print"),this); //
buraaazzzzzzzz
    printAction->setCheckable(true);                                          //
buraaazzzzzzzz
    printAction->setShortcut(tr("Ctrl+P"));                                   //
buraaazzzzzzzz
    connect(printAction, SIGNAL(triggered()),this,SLOT(printScene()));  //
buraaazzzzzzzz



    underlineAction = new QAction(QIcon(":/images/underline.png"),
tr("Underline"), this);
    underlineAction->setCheckable(true);
    underlineAction->setShortcut(tr("Ctrl+U"));
    connect(underlineAction, SIGNAL(triggered()), this, SLOT(handleFontChange()));

    aboutAction = new QAction(tr("A&bout"), this);
    aboutAction->setShortcut(tr("F1"));
    connect(aboutAction, SIGNAL(triggered()), this, SLOT(about()));

    copyAction = new QAction(QIcon(":/images/copy.png"), tr("C&opy"), this);
    copyAction->setShortcut(tr("Ctrl+C"));
    copyAction->setStatusTip(tr("Copy items from diagram"));
    connect(copyAction, SIGNAL(triggered()), this, SLOT(copyItem()));

    pasteAction = new QAction(QIcon(":/images/paste.png"), tr("P&aste"), this);
    pasteAction->setShortcut(tr("Ctrl+V"));
    pasteAction->setStatusTip(tr("Paste items from copyboard to diagram"));
    connect(pasteAction, SIGNAL(triggered()), this, SLOT(pasteItem()));

    cutAction = new QAction(QIcon(":/images/cut.png"), tr("C&ut"), this);
    cutAction->setShortcut(tr("Ctrl+X"));
    cutAction->setStatusTip(tr("Cut items from diagram"));
    connect(cutAction, SIGNAL(triggered()), this, SLOT(cutItem()));

    undoAction = new QAction(QIcon(":images/undo.png"), tr("U&ndo"), this);
    undoAction->setShortcut(tr("Ctrl+Z"));
    undoAction->setStatusTip(tr("Undo last operation"));
    connect(undoAction, SIGNAL(triggered()), this, SLOT(undo()));

    redoAction = new QAction(QIcon(":images/redo.png"), tr("R&edo"), this);
    redoAction->setShortcut(tr("Ctrl+Shift+Z"));
    redoAction->setStatusTip(tr("Redo last operation"));
    connect(redoAction, SIGNAL(triggered()), this, SLOT(redo()));

    groupAction = new QAction(QIcon(":images/group.png"), tr("G&roup"), this);
    groupAction->setStatusTip(tr("Group graphic items "));
    connect(groupAction, SIGNAL(triggered()), this, SLOT(groupItems()));

    ungroupAction = new QAction(QIcon(":images/ungroup.png"), tr("U&ngroup"),
this);
    ungroupAction->setStatusTip(tr("Ungroup graphic items"));
    connect(ungroupAction, SIGNAL(triggered()), this, SLOT(ungroupItems()));
}

void MainWindow::createMenus()
{
    fileMenu = menuBar()->addMenu(tr("&File"));
    fileMenu->addAction(exitAction);

    itemMenu = menuBar()->addMenu(tr("&Item"));
    itemMenu->addAction(copyAction);
    itemMenu->addAction(cutAction);
    itemMenu->addAction(pasteAction);
    itemMenu->addAction(deleteAction);
    itemMenu->addSeparator();
    itemMenu->addAction(undoAction);
    itemMenu->addAction(redoAction);
    itemMenu->addSeparator();
    itemMenu->addAction(groupAction);
    itemMenu->addAction(ungroupAction);
    itemMenu->addSeparator();
    itemMenu->addAction(toFrontAction);
    itemMenu->addAction(sendBackAction);
    itemMenu->addAction(printAction);

    aboutMenu = menuBar()->addMenu(tr("&Help"));
    aboutMenu->addAction(aboutAction);
}

void MainWindow::createToolbars()
{
    editToolBar = addToolBar(tr("Edit"));
    editToolBar->addAction(copyAction);
    editToolBar->addAction(cutAction);
    editToolBar->addAction(pasteAction);
    editToolBar->addAction(deleteAction);
    editToolBar->addAction(undoAction);
    editToolBar->addAction(redoAction);
    editToolBar->addAction(toFrontAction);
    editToolBar->addAction(sendBackAction);
    editToolBar->addAction(groupAction);
    editToolBar->addAction(ungroupAction);
    editToolBar->addAction(printAction);
    removeToolBar(editToolBar);
    addToolBar(Qt::LeftToolBarArea, editToolBar);
    editToolBar->show();

    fontCombo = new QFontComboBox();
    connect(fontCombo, SIGNAL(currentFontChanged(QFont)),
            this, SLOT(currentFontChanged(QFont)));
    fontCombo->setEditable(false);

    fontSizeCombo = new QComboBox;
    fontSizeCombo->setEditable(true);
    for (int i = 8; i < 30; i = i + 2)
        fontSizeCombo->addItem(QString().setNum(i));
    QIntValidator *validator = new QIntValidator(2, 64, this);
    fontSizeCombo->setValidator(validator);
    connect(fontSizeCombo, SIGNAL(currentIndexChanged(QString)),
            this, SLOT(fontSizeChanged(QString)));

    fontColorToolButton = new QToolButton;
    fontColorToolButton->setPopupMode(QToolButton::MenuButtonPopup);
    fontColorToolButton->setMenu(createColorMenu(SLOT(textColorChanged()),
Qt::black));
    textAction = fontColorToolButton->menu()->defaultAction();
    fontColorToolButton->setIcon(createColorToolButtonIcon(":/images/
textpointer.png", Qt::black));
    fontColorToolButton->setAutoFillBackground(true);
    connect(fontColorToolButton, SIGNAL(clicked()),
            this, SLOT(textButtonTriggered()));
```

```
//! [26]
    fillColorToolButton = new QToolButton;
    fillColorToolButton->setPopupMode(QToolButton::MenuButtonPopup);
    fillColorToolButton->setMenu(createColorMenu(SLOT(itemColorChanged()),
Qt::white));
    fillAction = fillColorToolButton->menu()->defaultAction();
    fillColorToolButton->setIcon(createColorToolButtonIcon(
                                 ":/images/floodfill.png", Qt::white));
    connect(fillColorToolButton, SIGNAL(clicked()),
            this, SLOT(fillButtonTriggered()));
//! [26]

    lineColorToolButton = new QToolButton;
    lineColorToolButton->setPopupMode(QToolButton::MenuButtonPopup);
    lineColorToolButton->setMenu(createColorMenu(SLOT(lineColorChanged()),
Qt::black));
    lineAction = lineColorToolButton->menu()->defaultAction();
    lineColorToolButton->setIcon(createColorToolButtonIcon(
                                 ":/images/linecolor.png", Qt::black));
    connect(lineColorToolButton, SIGNAL(clicked()),
            this, SLOT(lineButtonTriggered()));

    textToolBar = addToolBar(tr("Font"));
    textToolBar->addWidget(fontCombo);
    textToolBar->addWidget(fontSizeCombo);
    textToolBar->addAction(boldAction);
    textToolBar->addAction(italicAction);
    textToolBar->addAction(underlineAction);

    colorToolBar = addToolBar(tr("Color"));
    colorToolBar->addWidget(fontColorToolButton);
    colorToolBar->addWidget(fillColorToolButton);
    colorToolBar->addWidget(lineColorToolButton);

    QToolButton *pointerButton = new QToolButton;
    pointerButton->setCheckable(true);
    pointerButton->setChecked(true);
    pointerButton->setIcon(QIcon(":/images/pointer.png"));
    QToolButton *linePointerButton = new QToolButton;
    linePointerButton->setCheckable(true);
    linePointerButton->setIcon(QIcon(":/images/linepointer.png"));

    pointerTypeGroup = new QButtonGroup(this);
    pointerTypeGroup->addButton(pointerButton, int(DiagramScene::MoveItem));
    pointerTypeGroup->addButton(linePointerButton, int(DiagramScene::InsertLine));
    connect(pointerTypeGroup, SIGNAL(buttonClicked(int)),
            this, SLOT(pointerGroupClicked(int)));

    sceneScaleCombo = new QComboBox;
    QStringList scales;
    //scales << tr("50") << tr("75") << tr("100") << tr("125") << tr("150");
    scales <<tr("100");
    sceneScaleCombo->addItems(scales);
    sceneScaleCombo->setCurrentIndex(0);
    sceneScaleCombo->setEditable(false);
    QIntValidator *scaleValidator = new QIntValidator(1, 200, this);
    sceneScaleCombo->setValidator(scaleValidator);
    connect(sceneScaleCombo, SIGNAL(currentTextChanged(QString)),
            this, SLOT(sceneScaleChanged(QString)));
    QLabel* percentLabel = new QLabel(tr("%"), this);

    pointerToolbar = addToolBar(tr("Pointer type"));
    pointerToolbar->addWidget(pointerButton);
    pointerToolbar->addWidget(linePointerButton);
    pointerToolbar->addWidget(sceneScaleCombo);
    pointerToolbar->addWidget(percentLabel);
//! [27]

}
//! [27]

//! [28]
QWidget *MainWindow::createBackgroundCellWidget(const QString &text, const QString
&image)
{
    QToolButton *button = new QToolButton;
    button->setText(text);
    button->setIcon(QIcon(image));
    button->setIconSize(QSize(50, 50));
    button->setCheckable(true);
    backgroundButtonGroup->addButton(button);

    QGridLayout *layout = new QGridLayout;
    layout->addWidget(button, 0, 0, Qt::AlignHCenter);
    layout->addWidget(new QLabel(text), 1, 0, Qt::AlignCenter);

    QWidget *widget = new QWidget;
    widget->setLayout(layout);

    return widget;
}
//! [28]

//! [29]
QWidget *MainWindow::createCellWidget(const QString &text,
DiagramItem::DiagramType type)
{

    DiagramItem item(type, itemMenu);
    QIcon icon(item.image());

    QToolButton *button = new QToolButton;
    button->setIcon(icon);
    button->setIconSize(QSize(50, 50));
    button->setCheckable(true);
    buttonGroup->addButton(button, int(type));
    QGridLayout *layout = new QGridLayout;
    layout->addWidget(button, 0, 0, Qt::AlignHCenter);
    layout->addWidget(new QLabel(text), 1, 0, Qt::AlignCenter);

    QWidget *widget = new QWidget;
    widget->setLayout(layout);



    return widget;
}
//! [29]

//! [30]
QMenu *MainWindow::createColorMenu(const char *slot, QColor defaultColor)
{
    QList<QColor> colors;
    colors << Qt::black << Qt::white << Qt::red << Qt::blue << Qt::yellow;
    QStringList names;
    names << tr("black") << tr("white") << tr("red") << tr("blue")
          << tr("yellow");

    QMenu *colorMenu = new QMenu(this);
    for (int i = 0; i < colors.count(); ++i) {
        QAction *action = new QAction(names.at(i), this);
        action->setData(colors.at(i));
        action->setIcon(createColorIcon(colors.at(i)));
        connect(action, SIGNAL(triggered()), this, slot);
        colorMenu->addAction(action);
        if (colors.at(i) == defaultColor)
            colorMenu->setDefaultAction(action);
    }
    return colorMenu;
}
//! [30]

//! [31]
QIcon MainWindow::createColorToolButtonIcon(const QString &imageFile, QColor
color)
{
    QPixmap pixmap(50, 80);
    pixmap.fill(Qt::transparent);
    QPainter painter(&pixmap);
    QPixmap image(imageFile);
    // Draw icon centred horizontally on button.
    QRect target(4, 0, 42, 43);
    QRect source(0, 0, 42, 43);
    painter.fillRect(QRect(0, 60, 50, 80), color);
    painter.drawPixmap(target, image, source);

    return QIcon(pixmap);
}
```

```
//! [31]

//! [32]
QIcon MainWindow::createColorIcon(QColor color)
{
    QPixmap pixmap(20, 20);
    QPainter painter(&pixmap);
    painter.setPen(Qt::NoPen);
    painter.fillRect(QRect(0, 0, 20, 20), color);

    return QIcon(pixmap);
}

QList<QGraphicsItem*> MainWindow::cloneItems(const QList<QGraphicsItem*>& items) {
    QHash<QGraphicsItem*, QGraphicsItem*> copyMap;
    foreach (QGraphicsItem* item, items) {
        if (item->type() == DiagramItem::Type) {
            copyMap[item] = qgraphicsitem_cast<DiagramItem*>(item)->clone();
        } else if (item->type() == DiagramTextItem::Type) {
            copyMap[item] = qgraphicsitem_cast<DiagramTextItem*>(item)->clone();
        }
    }

    // connect DiagramItem with new arrow
    foreach (QGraphicsItem* item, items) {
        if (item->type() == Arrow::Type) {
            Arrow* arrow = qgraphicsitem_cast<Arrow*>(item);
            DiagramItem* copiedStartItem =
                    qgraphicsitem_cast<DiagramItem*>(copyMap.value(arrow-
>startItem(), nullptr));
            DiagramItem* copiedEndItem =
                    qgraphicsitem_cast<DiagramItem*>(copyMap.value(arrow-
>endItem(), nullptr));

            if (copiedStartItem == nullptr || copiedEndItem == nullptr) continue;

            Arrow* newArrow = new Arrow(copiedStartItem, copiedEndItem, nullptr);
            newArrow->setColor(arrow->getColor());

            copiedStartItem->addArrow(newArrow);
            copiedEndItem->addArrow(newArrow);
            newArrow->setZValue(-1000.0);

            copyMap[item] = newArrow;
        }
    }
    return copyMap.values();
}
//! [32]
```