Lecture Qt012
Drag and Drop

Instructor: David J. Coe

CPE 353 – Software Design and Engineering

Department of Electrical and Computer Engineering

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Outline

- Overview
- MIME
- Drag and Drop
- Drag and Drop Dialog
- DragLabel Class
- DropLabel Class
- Key Points

Overview

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

- Drag and Drop
  - The ability to use the mouse to transfer information between
    - Between two widgets within the same program
    - Between two applications

MIME

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

- Qt uses **MIME** data encoding for transfer via Drag and Drop

- **MIME**
  - **M**ultipurpose **I**nternet **M**ail **E**xtensions

QMimeData

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

- **QMimeData** class
  - Serves as container for the data
  - One may wish to drag-and-drop different types of data (text, images, etc.)
  - **QMimeData** supports multiple MIME data types
    - Example: PNG image has MIME type image/png
    - Other types include: text/plain, text/html, text/uri-list
- Qt Drag and Drop is implemented via the event handling mechanism

Drag and Drop

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

- A left mouse press is the typical initiation of a drag-and-drop action
  - Need to modify mouse press event handler of the data source to prepare widget contents for a possible drag event

Drag and Drop

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

- Must also prepare target widget to receive (or reject) the drop
  - A widget is a possible target of the drop as soon as the cursor is placed over the widget
  - The acceptDrops boolean flag determines whether a drop may proceed
    - true implies a drop **may** be possible
    - Use setAcceptDrops(bool …) to modify the value of this flag

Drag and Drop

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

- Must also prepare target widget to receive (or reject) the drop [continued...]
  - A drop event handler must be in place to extract the relevant information from the dragged **QMimeData** object and place it in the target widget

Summary: Drag and Drop

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

- **Source Widget**
  - Reimplement **mousePressEvent()**
- **Target Widget**
  - Reimplement **dragEnterEvent()** and **dropEvent()**

Drag and Drop

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

- **void QLabel::mousePressEvent (QMouseEvent * event) [virtual protected]**
  - Called when a mouse button is pressed
- **void QWidget::dragEnterEvent (QDragEnterEvent * event) [virtual protected]**
  - Called when a drag is in progress and the mouse enters this widget
  - If the event is ignored, the widget won't receive any drag move events
- **void QWidget::dragLeaveEvent (QDragLeaveEvent * event) [virtual protected]**
  - Called when a drag is in progress and the mouse leaves this widget
- **void QWidget::dragMoveEvent (QDragMoveEvent * event) [virtual protected]**
  - Called if a drag is in progress, and when any of the following conditions occur:
    - the cursor enters this widget, the cursor moves within this widget, or a modifier key is pressed on the keyboard while this widget has the focus
- **void QWidget::dropEvent (QDropEvent * event) [virtual protected]**
  - Called when the drag is dropped on this widget

- Description from Qt Assistant. Additional details available in Drag and Drop docs.

Drag and Drop Dialog

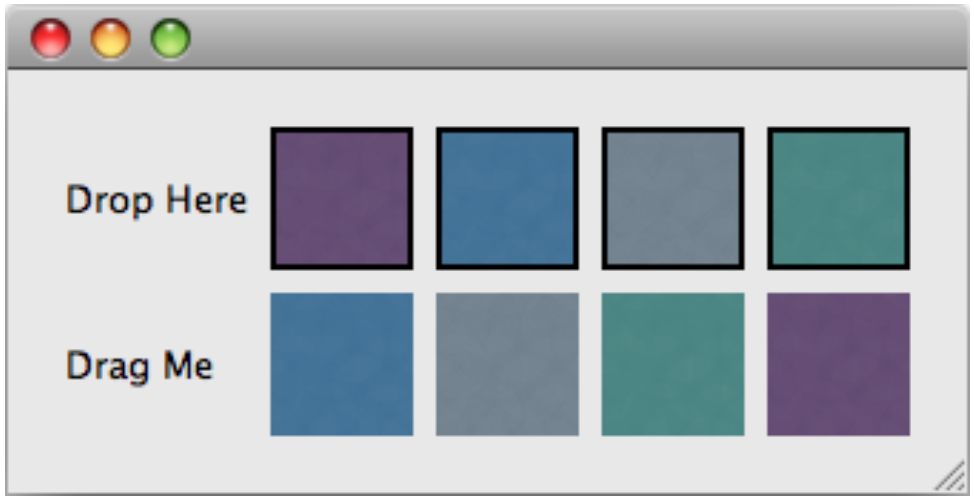
THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

```
#include <QApplication>
#include "dialog.h"

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);

    Dialog w;
    w.show();

    return a.exec();
}
```



Drag and Drop Dialog

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

```
#ifndef DIALOG_H
#define DIALOG_H
#include <QDialog>
#include "draglabel.h"
#include "droplabel.h"

class Dialog : public QDialog
{
    Q_OBJECT

public:
    Dialog(QWidget *parent = 0);
    ~Dialog();

private:
    DropLabel*  d1;         // Drop targets
    DropLabel*  d2;
    DropLabel*  d3;
    DropLabel*  d4;

    DragLabel*  s1;         // Drag sources
    DragLabel*  s2;
    DragLabel*  s3;
    DragLabel*  s4;

    QLabel*     d0;         // Text labels
    QLabel*     s0;

};

#endif // DIALOG_H
```

Drag and Drop Dialog

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

```
// dialog.cpp
#include "dialog.h"
#include "draglabel.h"
#include "droplabel.h"
#include <QVBoxLayout>
#include <QHBoxLayout>

Dialog::Dialog(QWidget *parent) : QDialog(parent)
{
    // Allocate and organize layouts
    QVBoxLayout* mainLayout = new QVBoxLayout(this);
    QHBoxLayout* dLayout = new QHBoxLayout;
    QHBoxLayout* sLayout = new QHBoxLayout;
    mainLayout->addLayout(dLayout);
    mainLayout->addStretch();
    mainLayout->addLayout(sLayout);

    // Populate destination layout
    d0 = new QLabel("Drop Here");
    d1 = new DropLabel;
    d2 = new DropLabel;
    d3 = new DropLabel;
    d4 = new DropLabel;
```

Drag and Drop Dialog

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

```
// dialog.cpp – continued

    dLayout->addWidget(d0);
    dLayout->addWidget(d1);
    dLayout->addWidget(d2);
    dLayout->addWidget(d3);
    dLayout->addWidget(d4);

    // Populate source layout
    s0 = new QLabel("Drag Me");
    s1 = new DragLabel(":/images/Picture1.png");
    s2 = new DragLabel(":/images/Picture2.png");
    s3 = new DragLabel(":/images/Picture3.png");
    s4 = new DragLabel(":/images/Picture4.png");

    sLayout->addWidget(s0);
    sLayout->addWidget(s1);
    sLayout->addWidget(s2);
    sLayout->addWidget(s3);
    sLayout->addWidget(s4);
}

Dialog::~Dialog()
{
    /* Empty autogenerated function */
}
```

DragLabel Class

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

```
// draglabel.h
#ifndef DRAGLABEL_H
#define DRAGLABEL_H

#include <QLabel>
#include <QString>
#include <QWidget>
#include <QString>

class DragLabel : public QLabel
{
    Q_OBJECT

public:
    DragLabel(const QString& path, QWidget* parent = 0);    // Constructor

protected:
    void mousePressEvent(QMouseEvent* event);               // Event handler

private:
    QString filename;                                       // Name of image file
};


#endif // DRAGLABEL_H
```

DragLabel Class

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

```
// draglabel.cpp
#include "draglabel.h"
#include <QPixmap>
#include <QMouseEvent>
#include <QImage>
#include <QMimeData>
#include <QDrag>
DragLabel::DragLabel(const QString & path, QWidget * parent) :
        QLabel(parent)    // Constructor
{
    filename = path;                 // Initialize member variable
    this->setFixedSize(50,50);       // Set label size
    setPixmap(QPixmap(filename));    // Make the image from the file the
                                     //  pixmap display in the label
}
```

DragLabel Class

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

```
// draglabel.cpp – continued

void DragLabel::mousePressEvent(QMouseEvent *event)// Reimplement event handler
{
    // If event is not NULL and left pushbutton pressed...
    if (event && event->button() == Qt::LeftButton)
    {
        // Allocate and initialize mimedata object
        QMimeData*  md = new QMimeData;
        QImage pic(filename);
        md->setImageData(pic);

        QDrag*  drag = new QDrag(this);     // Create a new drag object
        drag->setMimeData(md);              // associate mimedata with the drag

        if (pixmap())                       // If this label has a pixmap
        {
            QSize s = this->sizeHint();             // ...determine size
            drag->setPixmap(pixmap()->scaled(s));  // ...and use to scale drag
        }
        drag->start();                      // Initiate drag
    }
}
```

DropLabel Class

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

```
// droplabel.h
#ifndef DROPLABEL_H
#define DROPLABEL_H
#include <QtGui>
#include <QString>
#include <QWidget>
#include <QLabel>
class DropLabel : public Qlabel
{
    Q_OBJECT

public:
    DropLabel(QWidget* parent = 0);                    // Constructor

protected:
    void dragEnterEvent(QDragEnterEvent* event);       // Event handlers
    void dropEvent(QDropEvent* event);
};
#endif // DROPLABEL_H
```

DropLabel Class

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

```
// droplabel.cpp
#include "droplabel.h"
#include <QPixmap>

DropLabel::DropLabel( QWidget * parent) : QLabel(parent)
{
    this->setFrameStyle(QFrame::WinPanel);      // Show frame
    this->setFixedSize(50,50);                  // Set label size
    setAcceptDrops(true);                       // Accept drop events
}

void DropLabel::dragEnterEvent(QDragEnterEvent* event)
// Reimplement drag enter event handler
{
    // if event is not null and there is mimedata
    if (event && event->mimeData())
    {
        // Retrieve mime data from QDragEnterEvent
        const QMimeData*  md = event->mimeData();

        // Check to see if mime data includes an image
        if (md->hasImage())
        {
            event->acceptProposedAction();      // If so, accept the drag action
        }
    }
}
```

DropLabel Class

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

```
// droplabel.cpp - continued

void DropLabel::dropEvent(QDropEvent* event)
// Reimplement drop event handler
{
   QPixmap  pic;                        // Create empty pixmap
   if (event && event->mimeData())  // if event is not null and there is mimedata
   {
       // Retrieve mime data from QDropEvent
       const QMimeData*  md = event->mimeData();

       if (md->hasImage())              // if image is available, retrieve it
       {
           pic = md->imageData().value<QPixmap>();  // Retrieve image as a pixmap
       }
   }
   if (!pic.isNull())             // if an image has been retrieved, load into label
   {
       setPixmap(pic);            // Use pixmap for label
   }
}
```

Key Points

- Drag and drop may be used to transfer data between objects within an application or between applications
- Since you need to reimplement event handlers to enable the drag and drop mechanism, you will want to use inheritance to create customized data types that will be used as the source or destination of the drag-and-drop action