Using C++ and Qt in Practice

For students of HI 5323
“Image Processing”

Stefan Birmanns, Ph.D.
School of Health Information Sciences

http://biomachina.org/courses/processing/04.html

Outline

1. Files (.cpp, .h)
2. Preprocessor, Compiler, Linker
3. Shared Libraries
4. Makefiles and Qmake
5. Qt
6. Qt Designer
7. Demo

File Types

- .cpp, .c – Source code files
  - Definition of your program

```
int funcA(int i)     ←————————————  Definition
{
  return 5*i;
}

int funcA(int i);    ←————————————  Declaration
```

  - Before using a function/class you need to declare or define it

```
int i;               ←————————————  Definition
extern int i;        ←————————————  Declaration
```

Modules

- Use multiple .cpp/.c files to group functions and classes into modules
  - E.g. “fileio.c filter.c display.c”
- Code is easier to understand, easier to locate functions, etc.
- Compilation time gets reduced
  - Only the changed module needs to get recompiled
  - Works only if no dependencies between modules exist
    - If one changes the interface of module A, also the other modules need to get recompiled

- Module A needs to know about functions/classes provided by Module B
  - Header files
- Every class/function/object can only get defined once!

Definitions

- One time definition rule:
  - Every class/function/object must be defined only once!

  - File fileio.c

```
int x;
```

  - File filter.c

```
float x;  // error! X already defined
extern int x; // ok, just declaration
extern float x; // error! wrong type
int funcA() { float x; } // ok, different scope
class clA
{
  float x;// also ok, different scope
}
```

File Types

- .h – Header files
  - Define a common interface between all modules
  - Function and class declarations

    ```
    - int funcA(int i);
    - class B { B(int i); }
    ```

  - But no definitions!

    ```
    - int funcA(int i){ return i*5; }  // not in .h!
    - int global_i;  // not in .h!
    ```

  - Everything can only be defined once! Headers get included multiple times.

Macros

- Macros are keywords that are replaced by another text **before** compilation

```
#define MAX_COLORS 16
#define PRINT(S) printf(S);
```

- Macros are evil
  - They will change the code before compilation => error messages difficult to understand because compiled code is not what you see in your editor!
  - No type checking! `MAX_COLORS` has no type.
  - Better alternative are const variables:

    ```
    const int MAX_COLORS=16;
    ```

  - Macros make the source code difficult to understand
    - The actual source code is generated later by the preprocessor

- Do not use macros if you write a new program.

```
#include <X11/Xlib.h>
#include <unistd.h>
typedef long O; typedef struct                    { O b,f,u,s,c,a,t,e,d; } C;
Display *d; Window w; GC g; XEvent e;
char Q[] = "Level %d   Score %d", m[222];
#define N(r) (random()%(r))
#define U I[n++]=L[n]=l; n%=222
#define K c=-l.u; l=I[i]; l.t=0; c+=l.u
#define E l.e--&&!--L[l.e].d&&(L[l.e].t=3)
#define M(a,e,i,o) a[0]=e,(a[1]=i)&&XFillPolygon(d,w,g,(void*)a,o,1,1)
#define F return
#define R while(
#define Y if(l.t
                      O p                                           ,B,
                   D,A=6,Z                                         ,S=0,v=
                0,n=0,W=400                                       ,H=300,a[7]
              ={ 33,99, 165,                                     231,297,363} ;
            XGCValues G={ 6,0                                   ,~0L,0,1} ; short
           T[]={ 0,300,-20,0,4                                 ,-20,4,10,4,-5,4,5,
         4,-20,4,20,4,-5,4,5,4,                               -10,4,20},b[]={ 0,0,4,
        0,-4,4,-4,-4,4,-4,4,4} ;                             C L[222],I[222];dC(O x){
       M(T,a[x],H,12); } Ne(C l,O                           s) { l.f=l.a=1; l.b=l.u=s;
      l.t=16; l.e=0; U; } nL(O t,O                         a,O b,O x,O y,O s,O p){ C l;
     l.d=0; l.f=s; l.t=t; y-=l.c=b;                       l.e=t==2?x:p; x-=l.s=a;s=(x|1)
    %2*x; t=(y|1)%2*y; l.u=(a=s>t?s:                     t)>>9;l.a=(x<<9)/a;l.b=(y<<9)/a;
   U; } di(C I){ O p,q,r,s,i=222;C l;                   B=D=0; R i--){ l=L[i]; Y>7){ p=I.s
  -l.s>>9; q=I.c-l.c>>9; r=l.t==8?l.b:                 l.a; s=p*p+q*q; if(s<r*r||I.t==2&&s<
  26) F S+=10; s=(20<<9)/(s|1); B+=p*s;               D+=q*s; }} F 0; } hi(O x,O d){ O i=A;
 R i--&&(x<a[i]-d||x>a[i]+d)); F i; }      dL(){ O      c,r=0, i=222,h; C l; R i--){ l=L[i];
 Y){ r++;c=l.f; Y==3){c=l.u; l.t=0;     E; }R c--){--     l.u;h=l.c>>9; Y>7){XDrawArc(d,w,g,
(l.s>>9)-++l.a,h-l.a,l.a*2,l.a*2,0   ,90<<8); if(!l.u){    I[i].t-=8; l=I[i]; } } else Y==2)M
(b,l.s>>9,h,6); else XDrawPoint(d    ,w,g,(l.s+=l.a)>>9,    h=(l.c+=l.b)>>9); Y==4&&!l.u){ Ne
(l,20); K; } Y&&l.t<3&&(di(l)||h>    H)){ if(h>H&&(c=hi(    l.s>>9,25))>=0){ dC(c); a[c]=a[--
A]; }Ne(l,30); Y==1){ E;K; } else    c=l.t=0;} Y==1&&h<H    -75&&!N(p*77)){ do{ nL(1,l.s,l.c,
                                      N(W<<9),H<<9,1,i+
                                        1); I[i].d++;
                                           }R N(3)
                                        );         K;
                                       l.u=c; c=0; } Y
                                      ==2){ l.s+=l.a+B;
                                     l.a= (l.e-l.s)/((H+
                                    20-h)|1); l.c+=l.b+D;
                                   M(b,l.s>>9,l.c>>9,6); }
                                  } L[i]=l; } } F r; } J(){
                                 R A) { XFlush(d); v&&sleep(
                                3); Z=++v*10; p=50-v; v%2&&hi
                               ((a[A]=N(W-50)+25),50)<0 &&A++;
                              XClearWindow (d,w); for(B=0; B<A;
                             dC(B++)); R Z|dL()){ Z&&!N(p)&&(Z--
                            ,nL(1+!N(p),N(W<<9), 0,N(W<<9),H<<9,1
                           ,0)); usleep(p*200); XCheckMaskEvent(d,
                          4,&e)&&A&&--S&&nL(4,a[N(A)]<<9,H-10<<9,e.
                         xbutton.x<<9,e.xbutton.y<<9,5,0);}S+=A*100;
                             B=sprintf(m,Q,v,S); XDrawString(d,w
                                     ,g,W/3,H/2,m,B); } }
main ()
{
O i=2;
d=XOpenDisplay(0);
w=RootWindow(d,0);
R i--) XMapWindow(d,w=XCreateSimpleWindow(d,w,0,0,W,H,0,0,0));
XSelectInput(d,w,4|1<<15);
XMaskEvent(d,1<<15,&e);
g=XCreateGC(d,w,829,&G);
}
```

http://www.ioccc.org

Compilation

- Generation of executable:
  1. Preprocessing
     - Merges all included header files and the .c/.cpp file
     - Applies all macros to the source file
  2. Compiling
     - Generates machine code for single source files
     - Generates object files (.o or .obj)
     - Generates syntax errors if code is not correct
  3. Linking
     - Links all the .o files together
     - Connects the point where a function is called with the function itself
     - Generates error messages if references cannot get resolved
     - Links executable to external libraries

Compilation

- Compiler is the front-end to all three programs

```
gcc hello.c –o hello
```

- Invokes preprocessor, compiler and linker
- In case of multiple modules:

```
gcc –c fileio.c –o fileio.o
gcc –c filter.c –o filter.o
gcc –o program fileio.o filter.o
```

- First two steps preprocess and compile the modules
- The last links everything together
- Linking with shared libraries:

```
gcc –o program fileio.o filter.o –lGL
```

- Other platforms:
  - `cl` – Visual C++ (Windows)
  - `cc` – some Unix workstations (e.g. SGI, SUN,…)

Shared Libraries

- External Libraries provide additional functionality
  - libc – printf, scanf, …
  - libgl – OpenGL (3D graphics)
  - libqt – Qt graphical user interface

- Static linking
  - Copy all the code of the library into the executable
    - Large executables
    - Long loading/startup times
    - Difficult to distribute (internet)
    - Don’t benefit if an external library gets improved/updated

Shared Libraries

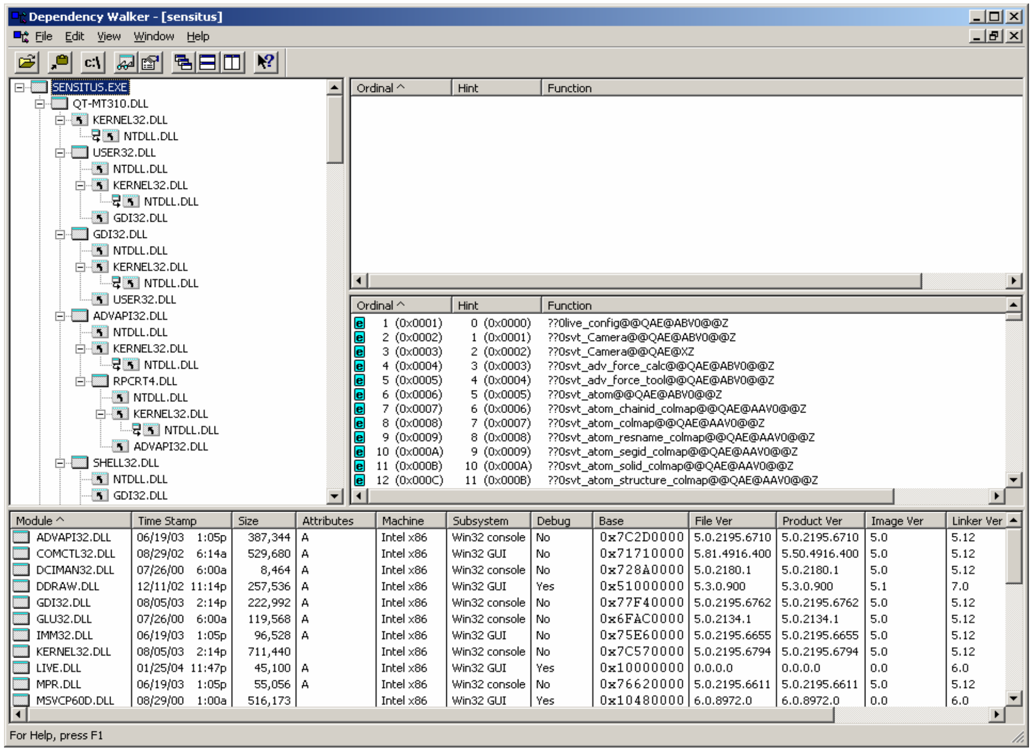
- Dynamic linking
  - The actual linking takes place during load time of the program
  - Executable code and library code are loaded from different files
    - filter (or filter.exe under Windows)
    - libqt.so (or qt.dll under Windows)
  - At startup all links between function calls and functions are created
  - If a reference cannot get resolved or a library file cannot get found, loading of the application will fail.
  - Libraries can also make use of another libraries
    - Recursive loading of all libraries
    - Not available under Windows

Shared Libraries

- Unix: ldd

```
ldd hello
  libc.so.6 -> /usr/lib/libc.so.6
  ld-linux.so.2 -> /lib/ld-linux.so.2
```

- Windows: Dependency Walker (http://dependencywalker.com/), free to use



```
SENSITUS.EXE
  QT-MT310.DLL
    KERNEL32.DLL
      NTDLL.DLL
    USER32.DLL
      NTDLL.DLL
      KERNEL32.DLL
        NTDLL.DLL
      GDI32.DLL
    GDI32.DLL
      NTDLL.DLL
      KERNEL32.DLL
        NTDLL.DLL
      USER32.DLL
    ADVAPI32.DLL
      NTDLL.DLL
      KERNEL32.DLL
        NTDLL.DLL
      RPCRT4.DLL
        NTDLL.DLL
        KERNEL32.DLL
          NTDLL.DLL
        ADVAPI32.DLL
    SHELL32.DLL
      NTDLL.DLL
      GDI32.DLL
```

Ordinal ^	Hint	Function
1 (0x0001)	0 (0x0000)	??0live_config@@QAE@ABV0@@Z
2 (0x0002)	1 (0x0001)	??0svt_Camera@@QAE@ABV0@@Z
3 (0x0003)	2 (0x0002)	??0svt_Camera@@QAE@XZ
4 (0x0004)	3 (0x0003)	??0svt_adv_force_calc@@QAE@ABV0@@Z
5 (0x0005)	4 (0x0004)	??0svt_adv_force_tool@@QAE@ABV0@@Z
6 (0x0006)	5 (0x0005)	??0svt_atom@@QAE@ABV0@@Z
7 (0x0007)	6 (0x0006)	??0svt_atom_chainid_colmap@@QAE@AAV0@@Z
8 (0x0008)	7 (0x0007)	??0svt_atom_colmap@@QAE@AAV0@@Z
9 (0x0009)	8 (0x0008)	??0svt_atom_resname_colmap@@QAE@AAV0@@Z
10 (0x000A)	9 (0x0009)	??0svt_atom_segid_colmap@@QAE@AAV0@@Z
11 (0x000B)	10 (0x000A)	??0svt_atom_solid_colmap@@QAE@AAV0@@Z
12 (0x000C)	11 (0x000B)	??0svt_atom_structure_colmap@@QAE@AAV0@@Z

Module ^	Time Stamp	Size	Attributes	Machine	Subsystem	Debug	Base	File Ver	Product Ver	Image Ver	Linker Ver
ADVAPI32.DLL	06/19/03 1:05p	387,344	A	Intel x86	Win32 console	No	0x7C2D0000	5.0.2195.6710	5.0.2195.6710	5.0	5.12
COMCTL32.DLL	08/29/02 6:14a	529,680	A	Intel x86	Win32 GUI	No	0x71710000	5.81.4916.400	5.50.4916.400	5.0	5.12
DCIMAN32.DLL	07/26/00 6:00a	8,464	A	Intel x86	Win32 console	No	0x728A0000	5.0.2180.1	5.0.2180.1	5.0	5.12
DDRAW.DLL	12/11/02 11:14p	257,536	A	Intel x86	Win32 GUI	Yes	0x51000000	5.3.0.900	5.3.0.900	5.1	7.0
GDI32.DLL	08/05/03 2:14p	222,992	A	Intel x86	Win32 console	No	0x77F40000	5.0.2195.6762	5.0.2195.6762	5.0	5.12
GLU32.DLL	07/26/00 6:00a	119,568	A	Intel x86	Win32 console	No	0x6FAC0000	5.0.2134.1	5.0.2134.1	5.0	5.12
IMM32.DLL	06/19/03 1:05p	96,528	A	Intel x86	Win32 GUI	No	0x75E60000	5.0.2195.6655	5.0.2195.6655	5.0	5.12
KERNEL32.DLL	08/05/03 2:14p	711,440		Intel x86	Win32 console	No	0x7C570000	5.0.2195.6794	5.0.2195.6794	5.0	5.12
LIVE.DLL	01/25/04 11:47p	45,100	A	Intel x86	Win32 GUI	Yes	0x10000000	0.0.0.0	0.0.0.0	0.0	6.0
MPR.DLL	06/19/03 1:05p	55,056	A	Intel x86	Win32 console	No	0x76620000	5.0.2195.6611	5.0.2195.6611	5.0	5.12
MSVCP60D.DLL	08/29/00 1:00a	516,173		Intel x86	Win32 GUI	Yes	0x10480000	6.0.8972.0	6.0.8972.0	0.0	6.0

For Help, press F1

Makefiles

- Compilation inconvenient if source code grows
    - Which modules have to get recompiled if I change module C?
- “Make” program steers the compilation process:
    - Makefiles define what the make program should do
    - Simple syntax:

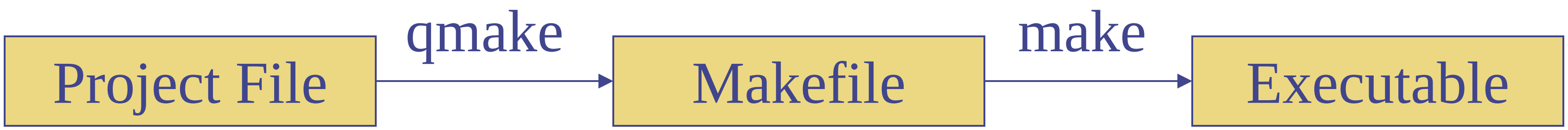
```
program: modA.o modB.o modC.o
  gcc –o program modA.o modB.o modC.o
```

    - “make” (“nmake” under Visual C++) on the command line will execute make program and start building the executable
    - Looks per default for “Makefile” in current directory
    - Default target: All
    - Additional targets like “clean” usually also defined

Makefiles

- But: makefile syntax depends on make program used
  - make/gmake - Linux
  - make – Windows (MinGW), SGI IRIX
  - nmake – Windows (Visual C++)

- Makefile generators
  - Generate platform dependent makefiles out of a platform independent project description file
  - cmake, tmake, qmake, imake, ant, …



qmake

- Part of Qt, but also useful for non-Qt projects
- Termed “tmake” in older Qt versions < 3.0
- Project file

```
TEMPLATE   =      app
CONFIG     +=     console qt
HEADERS    =      fileio.h filter.h
SOURCES    =      fileio.cpp filter.cpp main.cpp
TARGET     =      program
Win32:DEFINES+=   QT_DLL (Windows specific)
```

- Generate Makefile

```
qmake project.pro
make (or nmake under Visual C++)
```

- Per default also the “clean” target is defined

qmake

- `qmake –project` attempts to generate project file automatically

- Generated Makefile will automatically execute uic and compile/link the new files

```
 INTERFACES    = dialog.ui
```

- qmake also takes care of the special preprocessor “moc” for the signals and slots and also automatically compiles and links the new files

GUI Libraries

- Alternative graphical user interface libraries
  - Motif (old commercial unix library)
  - Xform/FLTK (old free unix library)
  - Win16 (Windows 3.1)
  - Win32 (Windows 95-present)
  - GTK (gimp/gnome)
  - TK (old free unix library)
- C++ Libraries to write applications with GUI:
  - MFC (Microsoft, based on Win32)
  - wxWindows (free platform independent library, based on GTK/Win32)
  - GTKmm (based on GTK)
  - Qt
- Important features: Portability, Price, Stability, Features

Qt

- Qt is a commercial software
  - Latest Unix/Linux versions free for academic use
  - Windows version relative inexpensive (ca. $700) for academic use
  - Free options for Windows:
    - Educational license for courses (need to apply)
    - Open-source version (see also session downloads)
      - Programs have to be published under GPL
      - Best with MinGW compiler (provided)
      - See next slide

- Qt is
  - Stable
  - Portable and well tested even on exotic platforms
  - Good documentation and tutorials available
  - A lot of sample code due to open source community

Qt & C++ Compilers under Windows

- Free open source version of Qt based on 4.0 (see session downloads)
- If you want to publish any software using this it needs to be open source under GPL license
- MinGW is a gcc (http://gcc.gnu.org) port to Windows, actually much better than what comes with Visual C++
- If one really wants to use Visual C++ one can add the missing configuration files for qmake:

  only a few config files are missing and one can take them e.g. from the Qt demo that is also available for free at trolltech.com.

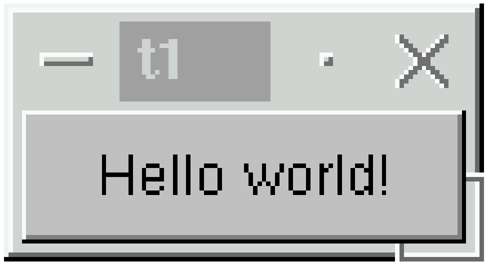
Qt

- C++ class library
  - GUI elements
    - Standard elements like buttons, sliders, …
    - Standard dialogs like “file open”, “choose color”, …
    - HTML renderer
    - OpenGL canvas
  - General purpose classes
    - Strings
    - File
    - Threads (create parallel programs for shared memory machines)
- Easy to use visual gui builder called “Qt Designer”

Qt

- “Hello World” example:

```
#include <qapplication.h>
#include <qpushbutton.h>
int main( int argc, char **argv ) {
  QApplication a( argc, argv );
  QPushButton hello( "Hello world!",
  0 );
  hello.resize( 100, 30 );
  a.setMainWidget( &hello );
  hello.show();
  return a.exec();
}
```



Signals/Slots

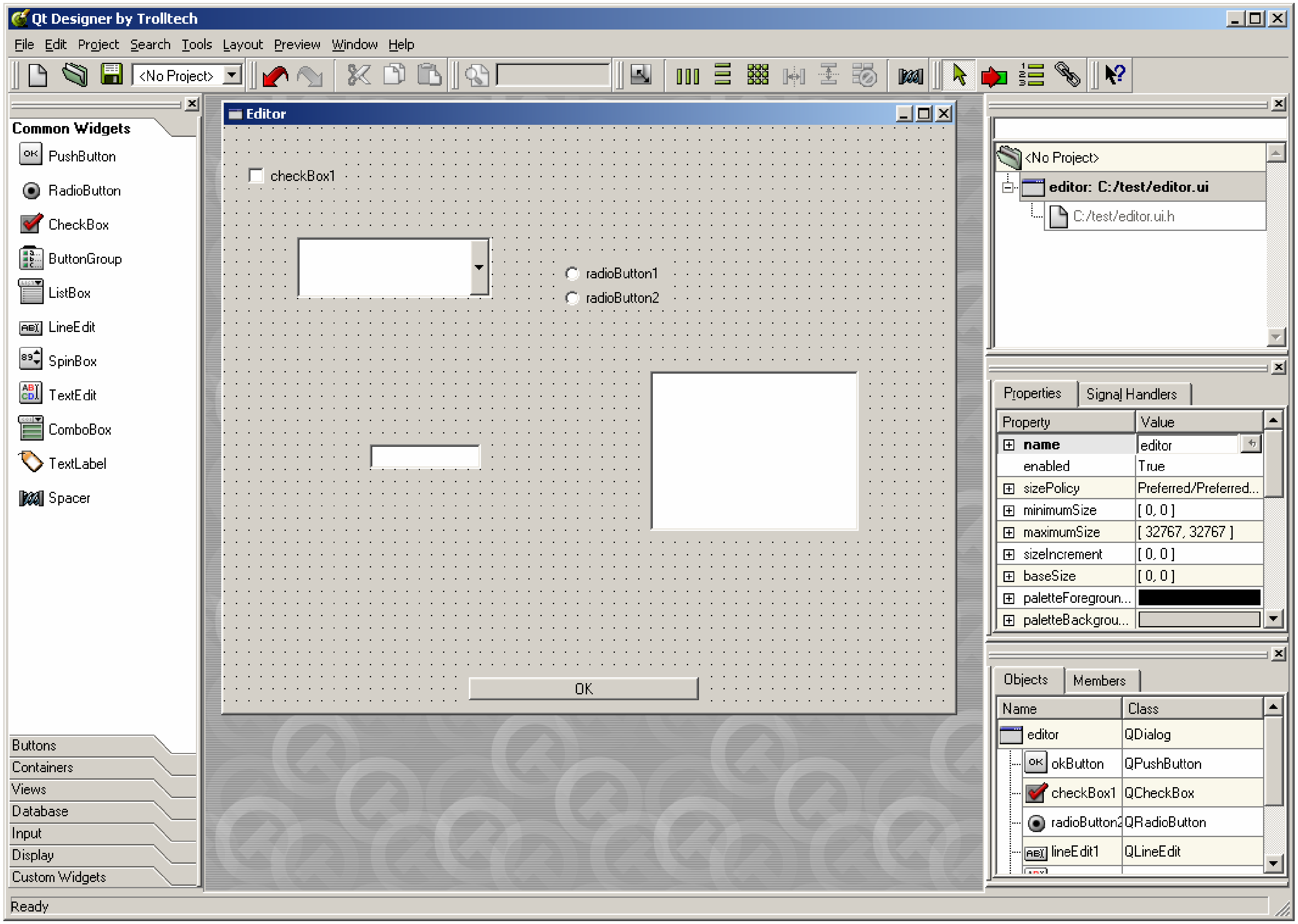
- Signals/Slots
  - Special new C++ language element
  - A function can generate a signal that gets caught by slots
  - A new preprocessor (“moc”) was developed to convert the extended C++ into regular C++
  - Every Qt class (and your own Qt related classes) feature new keywords:

```
class test
{
        Q_OBJECT
        …
  public slots:
        void load();
}
```

  - Slot can get connected to an object (like a button) that emits signals.
  - Please read documentation on http://www.trolltech.com

    → developer → documentation → overviews → signals and slots

Qt Designer



Property	Value
name	editor
enabled	True
sizePolicy	Preferred/Preferred...
minimumSize	[0, 0]
maximumSize	[32767, 32767]
sizeIncrement	[0, 0]
baseSize	[0, 0]
paletteForegroun...	■
paletteBackgrou...	□

Name	Class
editor	QDialog
okButton	QPushButton
checkBox1	QCheckBox
radioButton2	QRadioButton
lineEdit1	QLineEdit

Designer

- .ui files
  - Generated by Qt Designer
  - Code independent description of your dialog boxes
  - “uic” program generates C++ code for you, see demo
  - But: Don’t change the code, because it gets overwritten the next time you change you dialog box in the designer

Demo

- Designer
  - Create simple text editor dialog box (text field and ok button)
  - Change name to “editor”, caption to “Editor” and “OK”
  - Connect “clicked” signal of button with “accepted” slot of background.
  - Save as editor.ui in new directory
- Open a text editor of your choice
  - Create project file
  - Create file main.cpp
- Compile
  - Use qmake to generate Makefile
  - Use make to compile
- Example files and description: see class web site
- More documentation available with Qt

Image Histogram Example – Using C++ and Qt

Introduction

- The histogram is a simple and important tool in the digital image processing
- The gray-level histogram counts the number of pixels at each gray level
- It facilitates the optimization of digitizing parameters and of boundary thresholds
- It also helps in point operation, algebra operation and geometric operation

Introduction (cont.)

- This ***imageview*** program is implemented in Qt environment. It reads and decodes input images and calculates and plots the histogram of images
- We use birds.jpg as an input for the program testing.
- The QImage class reads the most common image file formats and can easily be expanded by plugins

Implementation

- The ***imageview*** program first reads an image file into a QImage object to construct an ImageView object, which is an instance of user-defined class of Image. The gray-level histogram is calculated in ImageView::histogram( ) function
- The image width and height is obtained from QImage::width() and QImage::height()
- Then QImage::scanLine( ) funtion returns a pointer point to the RGB value of each pixel by scanning through every pixel in the image

Implementation (cont.)

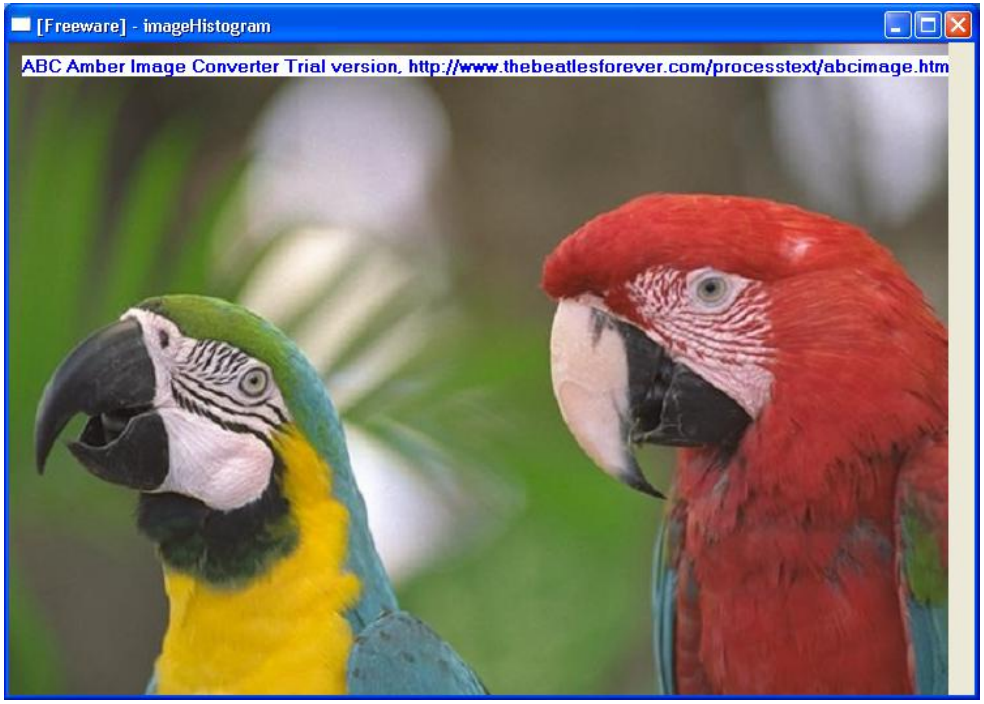
- The red, green and blue components are derived by the functions of qRed(), qGreen() and qBlue() functions. The corresponding gray level is calculated by the equation

gray level = (int) (0.299 * Red + 0.587 * Green + 0.114 * Blue + 0.5)

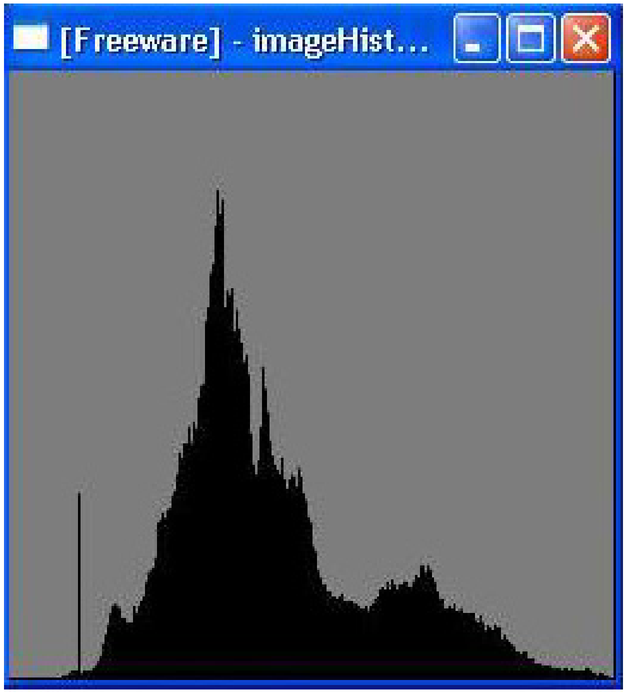
- The equation also considers the rounding of gray level
- At the end of the function an “ulong” array is returned that contains the pixel numbers within the range of 0 and 255
- The histogram array constructs a HistoView object that displays the gray-level histogram

Implementation (cont.)

- The following figure displays an image example and its histogram generated by the program.



ABC Amber Image Converter Trial version, http://www.thebeatlesforever.com/processtext/abcimage.htm



Program Files

- Header files
  - histoview.h
- CPP files
  - histoview.cpp
  - imageview.cpp
- Project file
  - imageHistogram.pro

Some Issues in Header File

- ***#ifdef* and *#ifndef***
  - *#ifdef* is used to include or omit statements from compilation depending of whether a macro name is defined or not
  - Often used to allow the same source module to be compiled in different environments (UNIX/ DOS/MVS), or with different options
  - *#ifndef* similar, but includes code when macro name is *not* defined

histoview.h

- Declare the class HistoView, its local variables, and some function to draw the histogram corresponding to an input image
  - Local variable: *protected: ulong m_iMax, m_aHist[256];*
  - Constructor: *HistoView(const ulong hist[256], QWidget *parent=0, const char *name=0);*
  - Function: *protected: virtual void paintEvent(QPaintEvent *);*

histoview.cpp

- Implement the class HistoView,
  - Define its constructor HistoView (*const ulong hist[256]*, *QWidget *parent*, *const char *name*)
  - Implement function *paintEvent(QPaintEvent *)* draw the histogram corresponding to an input image

imageview.cpp

- Implement the class ImageView
  - Local variables:
    - *const QImage m_oImage;*
    - *ulong hgram[256];*
  - Define its constructor *ImageView* (*const ulong hist[256]*, *QWidget *parent*, *const char *name*)
  - Implement function *ulong *histogram()* to calculate the graylevel histogram

imageview.cpp (cont.)

- Implement function *void paintEvent(QPaintEvent *)* display an input image
- Define function *void mousePressEvent (QMouseEvent *)* to close the window by clicking on image window

Acknowledgement

Wendy Xiong (Training Consultant 2004)

Help / Documentation

- Qt documentation comes with distribution
- Third party webpages and online forums like

  http://www.qtforum.org