QR Codes

Christian Grothoff

Berner Fachhochschule

March 2, 2018

Exam preview

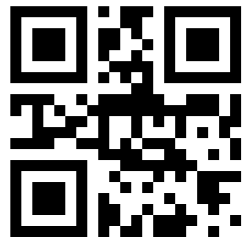
1. You may be asked questions about the project(s)
2. You may be asked questions about QR/GPS/GIS/LOC that exercises would help you answer

Experience:

Students doing all exercises rarely fail in my exams, students not doing any exercises very rarely pass in my exams.

Also, for this course, I do not care what language you do your projects in, some languages are more suitable than others, but the choice is entirely yours!

A QR code



- Created in 1994 by Denso-Wave (subsidiary of Toyota)
- Use is license-free
- The squares are called *modules*
- QR code must be surrounded by 4-module wide quiet zone

Position symbols and their borders

These are used by the device attempting to read the symbol. They fix the rotation and basic dimensions.

Alignment symbols

The number of these grows with the size of the symbol.
They help correct for perspective, curvature, and other distortion.

Timing arrays

Dotted lines connecting the alignment symbols:

These help to determine the dimensions of the symbol.

Dark module

- Always back
- Next to bottom left finder
- Coordinate $(4V + 4, 8)$

Version information

This is not a format version, but determines the *dimensions* of the image.

The size of version V is $N \times N$ with $N = 17 + 4V$.

Format information

Level of error correction chosen, and the index of the mask laid over the original message. Stored in several places.

Format information

EC level Determines error correction capability (Reed-Solomon is used)

Mask Laid over the symbol to minimize undesirable traits ("penalty rules")

Mode Determines type of input for efficient encoding (number, alpha-numeric, text, etc.)

L (7%)

M (15%)

Q (25%)

H (30%)

Why EC?

Modes

- Numeric
- Alpha-Numeric (case-insensitive, with $%*+-,/: and space)
- Byte
- Kanji (Japanese company!)
- Extended channel interpreteation
- Structured append mode
- FNC1

Maximum sizes

A 40-L QR code (177x177, 7% EC) can store:

Numeric	7089 characters
Alpha-Numeric	4296 characters
Byte	2953 characters
Kanji	1817 characters

Mask penalty rules

- Group of five or more same-colored modules in a row
- 2×2 area of same-colored modules
- Large penalty for patterns similar[1] to finder patterns
- Penalty proportional to black-white ratio imbalance

[1] 01000101111 or 11110100010

Think first!

https://www.youtube.com/watch?v=V2rVYvylvZc (11'2011)

Create QR code

```
#include <qrencode.h>

QRinput * qri;
QRcode *qrc;

qri = QRinput_new2 (0, QR_ECLEVEL_M);
QRinput_append (qri,
                QR_MODE_AN,
                strlen (text),
                (unsigned char*) text);
qrc = QRcode_encodeInput (qri);
```

Create pixel buffer

```
#include <gdk-pixbuf/gdk-pixbuf.h>

unsigned int size = qrc->width * scale;
size += 8 - (size % 8);

GdkPixbuf *pb = gdk_pixbuf_new (...);

guchar *pixels = gdk_pixbuf_get_pixels (pb);

int n_channels = gdk_pixbuf_get_n_channels (pb);
```

Set bits in pixel buffer

```
for (unsigned int x=0;x<size;x++)
  for (unsigned int y=0;y<size;y++) {
    off = (x * qrc->width / size) +
       (y * qrc->width / size) * qrc->width;
    for (int c = 0; c < n_channels; c++)
      pixels[(y * size + x) * n_channels + c]
        = (0 == (qrc->data[off] & 1)) ? 0xFF : 0;
  }
```

Use pixel buffer for Gtk+ image

```
GtkImage *image;

image = GTK_IMAGE (...);
gtk_image_set_from_pixbuf (image, pb);
```

Free resources

```
QRcode_free (qrc);
QRinput_free (qri);
g_object_unref (pb);
```

Exercise

- Write a C program that generates a QR code
- The input text should be taken from command-line ("argv[1]")
- Write the image output as X or spaces to the console, one character per pixel
- Bonus: use `getopt` to support command-line options for the various QR encoder options!

QR Codes in LaTeX

```
\usepackage{pspicture}
\usepackage{pst-barcode}
\usepackage{auto-pst-pdf}

 \begin{center}
    \leavevmode
    \begin{pspicture}(15mm,15mm)
        \psbarcode{text here}{eclevel=Q}{qrcode}
  \end{pspicture}
  \end{center}
```

Scanning QR codes

We will use zbar:

```
import sys
import getopt
import subprocess
from sys import argv
try:
        import zbar
except ImportError as e:
        print('Cannot run, please install zbar-python')
        sys.exit (1)
```

Opening the camera

Open camera:

```
device = '/dev/video0'
proc = zbar.Processor()
proc.parse_config('enable')
proc.init(device)
```

Read a QR code

```
try:
    proc.process_one()
except Exception as e:
    # Window was closed without finding code
    exit (1)
```

Use results

```
for symbol in proc.results:
  print('Found ', symbol.type, ' symbol ', '"%s"' \
    % symbol.data)
```

Exercise

- Implement logic to read QR code
- Use `https://www.thonky.com/qrcode/` to generate and print different QR codes (text length, error correction, output size)
- Test your reader against examples

Android: Reacting to URLs via schema registration

In you manifest, use:

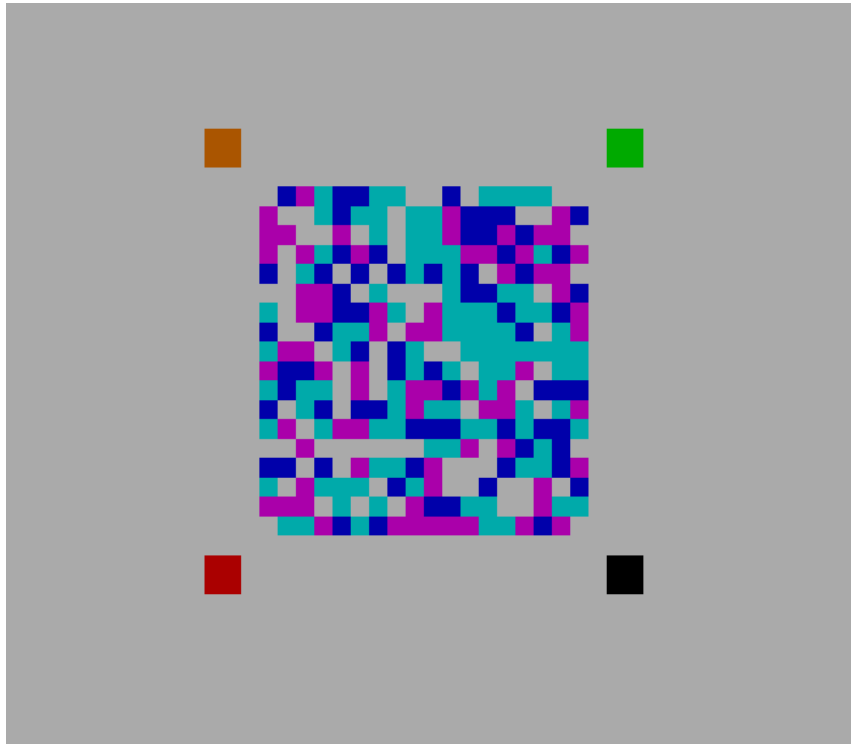
```
<intent-filter>
  <action android:name="android.intent.action.VIEW" />
  <category android:name="android.intent.category.DEFAULT" />
  <category android:name="android.intent.category.BROWSABLE" />
  <data android:scheme="mailto" />
  <data android:scheme="http" android:host="example.com" />
</intent-filter>
```

to hook the `mailto` schema and “http://example.com/”.

Android: Barcode Detection

https://codelabs.developers.google.com/codelabs/bar-codes/

Further reading



Acknowlegements

This presentation used material from:

- `http://www.ams.org/samplings/feature-column/fc-2013-02`
- `https://www.thonky.com/qrcode-code-tutorial/`
- `https://gnunet.org/`