KWIn went XCB

Martin Gräßlin
mgraesslin@kde.org
Blue Systems

2014 X.Org Developer's Conference

October 2014
Who am I?

Martin Gräßlin

- e-mail: mgraesslin@kde.org
- IRC: mgraesslin on freenode
- KWin maintainer
- Works for Blue Systems GmbH with focus on KWin
Facts about KWin

- Window Manager and Compositor for Plasma by KDE
- git://anongit.kde.org/kwin.git
- Used since KDE 2.0
- In development since 1999
- > 12400 commits
- > 360 contributors
- 127000 lines of code
Agenda

Why?

Obstacles during Porting
  Documentation
  XCB-ICCCM
  util-renderutil
  Sync Extension
  XEvent Porting
  Unit Tests

C++ and XCB
  XCB API wrapper
  Window

Current State
Agenda

Why?

Obstacles during Porting
  Documentation
  XCB-ICCCM
  util-renderutil
  Sync Extension
  XEvent Porting
  Unit Tests

C++ and XCB
  XCB API wrapper
  Window

Current State
Porting Notes

Note: The native events that can be filtered this way depend on the QPA backend chosen at runtime. On X11, xEvents are replaced with xcb_generic_event_t due to the switch to XCB, which requires porting the application code to XCB as well.
Qt 5 API Changes

Broken Assumptions

- Qt on Linux implies X11
- QWidget == X11 Window
- QPixmap == X11 Pixmap
- QCursor == X11 Cursor
- QRegion == X11 Region

Reference

“We are the 1 %” - Akademy talks 2013:
The state before Porting

Multiple Modules

- 1500 lines of huge switch statement in KWin
- 4500 lines for NETWM handling in KWindowSystem
- XGetWindowAttributes per window on KWin startup
- up to 15 XGetWindowProperty in NETWM during KWin startup
- up to 27 XGetWindowProperty in NETWM for each managed window in KWin
- Several properties are read multiple times in KWin and NETWM (e.g. three times XGetWMHints)
- 0 % test coverage
Agenda

Why?

**Obstacles during Porting**
- Documentation
- XCB-ICCCM
- util-renderutil
- Sync Extension
- XEvent Porting
- Unit Tests

**C++ and XCB**
- XCB API wrapper
- Window

**Current State**
Agenda

Why?

Obstacles during Porting
   Documentation
   XCB-ICCCM
   util-renderutil
   Sync Extension
   XEvent Porting
   Unit Tests

C++ and XCB
   XCB API wrapper
   Window

Current State
Improved!

Here I wanted to rant about the pity state of XCB API docs. But when writing this presentation I noticed that it significantly improved!
Extensions - Still need improvement

/**
 * @param c The connection
 * @return A cookie
 *
 * Delivers a request to the X server.
 *
 */

xcb_void_cookie_t
xcb_damage_create (xcb_connection_t *c /**< */,
                    xcb_damage_damage_t damage /**< */,
                    xcb_drawable_t drawable /**< */,
                    uint8_t level /**< */);
Extensions

Where are they?

- Try to Google it
- http://www.x.org/wiki/Documentation/?
- x.org → X11R7.7 → http://www.x.org/releases/X11R7.7/ → Documentation for X11R7.7

Where is a PDF?

- XRender
- Composite
- Damage
- XFixes
- XInput

Only XLib?

X Synchronization Extension Library
Annotated X protocol?

Who needs this?

- PolyPlane
- PolyLine
- PolySegment
- PolySegment
- and so on and on

Suggestion

Provide an annotated X protocol document with legacy stuff removed and important extensions added.

What one would need, though:

- Shm extension to do client side rendering
- Render extension to do server side rendering
- Input2 for proper input handling
- and so on and on
Agenda

Why?

Obstacles during Porting
- Documentation
- XCB-ICCCM
- util-renderutil
- Sync Extension
- XEvent Porting
- Unit Tests

C++ and XCB
- XCB API wrapper
- Window

Current State
XCB-ICCCM cannot reasonably be used

Split repository

- Used to be part of xcb-util
- Now in util-wm
API of XCB-ICCCM changed

Example: Old

```c
xcb_void_cookie_t xcb_set_wm_name_checked(xcb_connection_t *c,
                                         xcb_window_t window,
                                         xcb_atom_t encoding,
                                         uint8_t format,
                                         uint32_t name_len,
                                         const char *name);
```

Example: New

```c
xcb_void_cookie_t xcb_icccm_set_wm_name_checked(xcb_connection_t *c,
                                               xcb_window_t window,
                                               xcb_atom_t encoding,
                                               uint8_t format,
                                               uint32_t name_len,
                                               const char *name);
```
The Problem with that

PROBLEM!
Distributions still don’t ship it!

Consequences

▶ Developer needs to choose between old or new API
▶ Distributions either provide old or new API
▶ ifdef hell?

Example usage in KWin’s config-kwin.h.cmake

```cmake
#cmakedefine XCB_ICCCM_FOUND 1
#ifndef XCB_ICCCM_FOUND
#define XCB_ICCCM_WM_STATE_WITHDRAWN 0
#define XCB_ICCCM_WM_STATE_NORMAL 1
#define XCB_ICCCM_WM_STATE_ICONIC 3
#endif
```
Agenda

Why?

Obstacles during Porting
- Documentation
- XCB-ICCCM
- util-renderutil
- Sync Extension
- XEvent Porting
- Unit Tests

C++ and XCB
- XCB API wrapper
- Window

Current State
Fun with C++

```cpp
#ifndef XCB_RENDERUTIL
#define XCB_RENDERUTIL
#include <xcb/render.h>
// <snip>
xcb_render_pictforminfo_t *
xcb_render_util_find_format (const xcb_render_query_pict_formats_reply_t *formats,
unsigned long mask,
const xcb_render_pictforminfo_t *template,
int count);
// <snip>
#endif /* XCB_RENDERUTIL */
```

Note
This is fixed as of commit 8d15acc45a47dc4c922eee5b99885db42bc62c17 (July 2013) after seven years!
Qt has a workaround

// 'template' is used as a function argument
// name in xcb_renderutil.h
#define template template_param
// extern "C" is missing too
extern "C" {
#include <xcb/xcb_renderutil.h>
}
#undef template

Source: qtbase/src/plugins/platforms/xcb/qxcbimage.cpp
Agenda

Why?

Obstacles during Porting
- Documentation
- XCB-ICCCM
- util-renderutil
- Sync Extension
- XEvent Porting
- Unit Tests

C++ and XCB
- XCB API wrapper
- Window

Current State
XCB port for sync extension is broken

commit e6a246e50e62cbbcba33d0e1d2371e69e6e089383
Author: Louis-Francis Ratte-Boulianne
Date: Tue Jul 2 19:21:40 2013 +0100

sync: Change value list param of CreateAlarm and ChangeAlarm into switch

Values for "Value" and "Delta" fields are 64-bit that couldn’t be passed through a regular value list/mask.

Signed-off-by: Louis-Francis Ratte-Boulianne
Signed-off-by: Peter Harris
Sync in KWin

Reverted back to XLib.
Agenda

Why?

Obstacles during Porting
- Documentation
- XCB-ICCCM
- util-renderutil
- Sync Extension
- XEvent Porting
- Unit Tests

C++ and XCB
- XCB API wrapper
- Window

Current State
Code example from KWin

```c
if (Client* c = findClient(WindowMatchPredicate(e->xany.window))) {
    if (c->windowEvent(e))
        return true;
} else if (Client* c = findClient(WrapperIdMatchPredicate(e->xany.window))) {
    if (c->windowEvent(e))
        return true;
} else if (Client* c = findClient(FrameIdMatchPredicate(e->xany.window))) {
    if (c->windowEvent(e))
        return true;
} else if (Client *c = findClient(InputIdMatchPredicate(e->xany.window))) {
    if (c->windowEvent(e))
        return true;
} else if (Unmanaged* c = findUnmanaged(WindowMatchPredicate(e->xany.window)))
    if (c->windowEvent(e))
        return true;
}```
typedef struct {
    int type;
    unsigned long serial;
    Bool send_event;
    Display *display;
    Window window;
} XAnyEvent;

The `window` member is set to the window that is most useful to toolkit dispatchers.
What is it on XCB?

???

Oh fuck

- Could be handling any event
- Some event types have multiple window elements
- Some have an "event" window, but is that the most useful to toolkit dispatchers?
 Same code today

```c
const xcb_window_t eventWindow = findEventWindow(e);
if (eventWindow != XCB_WINDOW_NONE) {
    if (Client* c = findClient(Predicate::WindowMatch, eventWindow)) {
        if (c->windowEvent(e))
            return true;
    } else if (Client* c = findClient(Predicate::WrapperIdMatch, eventWindow)) {
        if (c->windowEvent(e))
            return true;
    } else if (Client* c = findClient(Predicate::FrameIdMatch, eventWindow)) {
        if (c->windowEvent(e))
            return true;
    } else if (Client *c = findClient(Predicate::InputIdMatch, eventWindow)) {
        if (c->windowEvent(e))
            return true;
    }
    // <snip>
}
```
findEventWindow

static xcb_window_t findEventWindow(xcb_generic_event_t *event)
{
    const uint8_t eventType = event->response_type & ~0x80;
    switch(eventType) {
    case XCB_KEY_PRESS:
    case XCB_KEY_RELEASE:
        return reinterpret_cast<xcb_key_press_event_t*>(event)->event;
    case XCB_BUTTON_PRESS:
    case XCB_BUTTON_RELEASE:
        return reinterpret_cast<xcb_button_press_event_t*>(event)->event;
    case XCB_MOTION_NOTIFY:
        return reinterpret_cast<xcb_motion_notify_event_t*>(event)->event;
    case XCB_ENTER_NOTIFY:
    case XCB_LEAVE_NOTIFY:
        return reinterpret_cast<xcb_enter_notify_event_t*>(event)->event;
    case XCB_FOCUS_IN:
    case XCB_FOCUS_OUT:
        return reinterpret_cast<xcb_focus_in_event_t*>(event)->event;
    case XCB_EXPOSE:
        return reinterpret_cast<xcb_expose_event_t*>(event)->window;
    // <snip>
    default:
        return XCB_WINDOW_NONE;
    }
}

Agenda

Why?

Obstacles during Porting
- Documentation
- XCB-ICCCM
- util-renderutil
- Sync Extension
- XEvent Porting
- Unit Tests

C++ and XCB
- XCB API wrapper
- Window

Current State
XLib based code base is not unit tested

Having tests before porting would be useful!
Running tests against X11 is difficult

Problems

- Some unit tests need a Window Manager, some don’t
- Xvfb is run for complete test session
- Our CI-system cannot use KWin
- CI-system uses openbox:
  - Our tests are able to crash openbox
  - KDE-specific features cannot be tested with openbox
  - Tests may pass locally with KWin but fail on CI
- Xvfb doesn’t support XRandR
- Mocking XCB is too much work
- Tests interacting with WM are fragile
Agenda

Why?

Obstacles during Porting
  Documentation
  XCB-ICCCM
  util-renderutil
  Sync Extension
  XEvent Porting
  Unit Tests

C++ and XCB
  XCB API wrapper
  Window

Current State
Agenda

Why?

Obstacles during Porting
- Documentation
- XCB-ICCCM
- util-renderutil
- Sync Extension
- XEvent Porting
- Unit Tests

C++ and XCB
- XCB API wrapper
- Window

Current State
auto cookie = xcb_get_geometry_unchecked(connection(), window());
auto reply = xcb_get_geometry_reply(connection(), cookie, nullptr);
if (reply) {
    QRect geo(reply->x, reply->y, reply->width, reply->height);
    // do something with geo
    if (reply->depth == 0) {
        // this would leak!
        return;
    }
    free(reply);
}
free is evil!

For a C++ developer

- free exceeds language barrier
- C++ uses delete
- C++ is memory managed

```cpp
template <typename T> using ScopedCPointer =
    QScopedPointer<T, QScopedPointerPodDeleter>;

auto cookie = xcb_get_geometry_unchecked(connection(), window());
ScopedCPointer<xcb_get_geometry_reply_t> reply(
    xcb_get_geometry_reply(connection(), cookie, nullptr));
if (!reply.isNull()) {
    QRect geo(reply->x, reply->y, reply->width, reply->height);
    // do something with geo
    if (reply->depth == 0) {
        // this would no longer leak!
        return;
    }
}
```
More ways to leak

// early fetch
auto cookie = xcb_get_geometry_unchecked(connection(), window());
// do something different and hit error condition
if (foo) {
    // this would leak if we don’t discard
    return;
}
ScopedCPointer<xcb_get_geometry_reply_t> reply(
    xcb_get_geometry_reply(connection(), cookie, nullptr));
if (!reply.isNull()) {
    QRect geo(reply->x, reply->y, reply->width, reply->height);
    // do something with geo
    if (reply->depth == 0) {
        return;
    }
}
Resource Acquisition Is Initialization to the rescue

// early fetch
Xcb::WindowGeometry reply(w);
// do something different and hit error condition
if (foo) {
    // this doesn’t leak any more due to RAII
    return;
}
if (!reply.isNull()) {
    QRect geo = reply.rect();
    // do something with geo
    // do something with geo
    if (reply->depth == 0) {
        return;
    }
}
How does it work?

- Ctor performs xcb_get_geometryUnchecked
- xcb_get_geometry_reply delayed till reply is needed first time
- Dtor frees reply if fetched
- Dtor discards reply if not fetched
- implements operator− > () to behave like a xcb_get_geometry_reply_t* if needed
- No need to pass the xcb_connection_t* any more
With the help of C++11

```cpp
template <typename Reply, typename Cookie, typename... Args>
struct WrapperData
{
    typedef Reply reply_type;
    typedef Cookie cookie_type;
    typedef std::tuple<Args...> argument_types;
    typedef Cookie (*request_func)(xcb_connection_t*, Args...);
    typedef Reply *(*reply_func)(xcb_connection_t*,
                                Cookie,
                                xcb_generic_error_t**);

    static constexpr std::size_t argumentCount = sizeof...(Args);
};

Concrete Example

struct GeometryData : public WrapperData< xcb_get_geometry_reply_t,
                                         xcb_get_geometry_cookie_t,
                                         xcb_drawable_t >
{
    static constexpr request_func requestFunc = &xcb_get_geometry_unchecked;
    static constexpr reply_func replyFunc = &xcb_get_geometry_reply;
};
```
Continued

template<typename Data> class AbstractWrapper
{
public:
    typedef typename Data::cookie_type Cookie;
    typedef typename Data::reply_type Reply;
    virtual ~AbstractWrapper();
    AbstractWrapper &operator=(const AbstractWrapper &other);
    const Reply *operator->();
    bool isNull(); // and same as const
    operator bool(); // and same as const
    const Reply *data(); // and same as const
    bool isRetrieved() const;
    Reply *take();

protected:
    AbstractWrapper();
    explicit AbstractWrapper(WindowId window, Cookie cookie);
    explicit AbstractWrapper(const AbstractWrapper &other);
    void getReply();

private:
    bool m_retrieved;
    Cookie m_cookie;
    Reply *m_reply;
};
And finally

```cpp
template<typename Data, typename... Args>
class Wrapper<Data, xcb_window_t, Args...> : public AbstractWrapper<Data>
{
public:
    // skip some static asserts
    Wrapper() = default;
    explicit Wrapper(xcb_window_t w, Args... args)
        : AbstractWrapper<Data>(w,
            Data::requestFunc(connection(), w, args...)) {};
};

class WindowGeometry : public Wrapper<GeometryData, xcb_window_t>
{
public:
    WindowGeometry() : Wrapper<GeometryData, xcb_window_t>() {};
    explicit WindowGeometry(xcb_window_t window)
        : Wrapper<GeometryData, xcb_window_t>(window) {};
    inline QRect rect() {
        const auto g = data();
        if (!g)
            return QRect();
        return QRect(g->x, g->y, g->width, g->height);
    }
};
```
What it means for KWin development

Advantages

- Works with all request/reply combinations
- Compile-time checks for argument specifications
- No code duplication
- Hides the xcb data types and requests
- Hides async nature of xcb: async when we want it to be async, sync when we want it to be sync
- All requests under tests
- CamelCase instead of underscores
- Proper C++ namespacing instead of xcb_extension_foo_get_bar
To the extreme: reading properties

Features

▶ Provides validation for:
  ▶ type
  ▶ format

▶ cast to requested type
▶ handles arrays and single value
▶ format retrieved from requested type
▶ default types for error cases
▶ Simplified reading for QByteArrayList
Example usage for property reading

// property is an array of uint32_t
Xcb::Property property(false, id, atoms->kde_net_wm_shadow,
    XCB_ATOM_CARDINAL, 0, 12);
uint32_t *shadow = property.value<uint32_t*>();

// property is a single value which is interpreted as a boolean value
Xcb::Property property(false, window(), atoms->kde_skip_close_animation,
    XCB_ATOM_CARDINAL, 0, 1);
setSkipCloseAnimation(property.toBool());

// reading a string property to a QByteArray
QByteArray prop = Xcb::StringProperty(window(), atoms->activities);
Agenda

Why?

Obstacles during Porting
   Documentation
   XCB-ICCCM
   util-renderutil
   Sync Extension
   XEvent Porting
   Unit Tests

C++ and XCB
   XCB API wrapper
   Window

Current State
Mixing C++11, Qt and XCB

Let’s move a window

```cpp
void moveResizeWindow(WindowId window, const QRect &geometry) {
    const uint16_t mask = XCB_CONFIG_WINDOW_X
                          | XCB_CONFIG_WINDOW_Y
                          | XCB_CONFIG_WINDOW_WIDTH
                          | XCB_CONFIG_WINDOW_HEIGHT;

    const uint32_t values[] = {
        geometry.x(),
        geometry.y(),
        geometry.width(),
        geometry.height()
    };
    xcb_configure_window(connection(), window, mask, values);
}
```
What the compiler thinks of it

xcbutils.h: In function 'void moveResizeWindow(WindowId, const QRect&)':
xcbutils.h:1334:20: warning: narrowing conversion of '(& geometry)->QRect::x()' from 'int' to 'const uint32_t {aka const unsigned int}' inside { } [-Wnarrowing]
    geometry.x(),
    ^
xcbutils.h:1335:20: warning: narrowing conversion of '(& geometry)->QRect::y()' from 'int' to 'const uint32_t {aka const unsigned int}' inside { } [-Wnarrowing]
    geometry.y(),
    ^
xcbutils.h:1336:24: warning: narrowing conversion of '(& geometry)->QRect::width()' from 'int' to 'const uint32_t {aka const unsigned int}' inside { } [-Wnarrowing]
    geometry.width(),
    ^
xcbutils.h:1337:25: warning: narrowing conversion of '(& geometry)->QRect::height()' from 'int' to 'const uint32_t {aka const unsigned int}' inside { } [-Wnarrowing]
    geometry.height()
Mixing C++11, Qt and XCB, continued

How it has to look like

```cpp
void moveResizeWindow(WindowId window, const QRect &geometry)
{
    const uint16_t mask = XCB_CONFIG_WINDOW_X |
                           XCB_CONFIG_WINDOW_Y |
                           XCB_CONFIG_WINDOW_WIDTH |
                           XCB_CONFIG_WINDOW_HEIGHT;

    const uint32_t values[] = {
        static_cast<uint32_t>(geometry.x()),
        static_cast<uint32_t>(geometry.y()),
        static_cast<uint32_t>(geometry.width()),
        static_cast<uint32_t>(geometry.height())
    };

    xcb_configure_window(connection(), window, mask, values);
}
```

This is ugly!

We don’t want to have that all over the place.
RAII for windows

Advantages from RAII

- If we create the window, we also want it to be destroyed
- Simpler API
- Hides the xcb_connection_t*
- Hides the xcb_window_t
- Drop in replacement for any place expecting xcb_window_t

Example

```cpp
const uint32_t mask = XCB_CW_OVERRIDE_REDIRECT | XCB_CW_EVENT_MASK;
const uint32_t values[] = {
    true,
    XCB_EVENT_MASK_ENTER_WINDOW |
    XCB_EVENT_MASK_LEAVE_WINDOW
};
const QRect geometry(0, 0, 100, 100);
Window window(geometry, XCB_WINDOW_CLASS_INPUT_ONLY, mask, values);
window.map();
```
What the Window wrapper provides

- Unit tests
- Manage foreign windows
- Everything inline
- Convenient methods like what XLib provided, e.g.:
  - lower()
  - raise()
  - move(const QPoint&)
  - resize(const QSize&)
  - selectInput(uint32_t)
Agenda

Why?

Obstacles during Porting
  Documentation
  XCB-ICCCM
  util-renderutil
  Sync Extension
  XEvent Porting
  Unit Tests

C++ and XCB
  XCB API wrapper
  Window

Current State
Not everything is ported

Areas which have to remain on XLib

- XSync extension
- XCursor interaction
- GLX compositing backend
- EGL/X11 compositing backend

Not yet ported

- Reading Motif hints
- Reading of _NET_WM_OPAQUE_REGION
- One call to XGetWMNormalHints
- One call to XGetWMHints (basically ported, blocked by freeze)
- Usage of XFree86-VidModeExtension
- Keymap interaction
NETWM in KWindowSystem ported

Advantages

▶ Doesn’t block for reading properties
▶ KWin gets more properties from NETWM
▶ Mostly under tests nowadays:
  ▶ Line coverage: 83 %
  ▶ Branch coverage: 72 %
Questions?