Maemomm
Maemo with C++ and a Gtkmm flavour

David King, davidk@openismus.com

Openismus GmbH
https://garage.maemo.org/projects/maemomm/

10th October 2009 / N800 room
1. Maemomm Overview
   - C++ bindings
   - Advantages
   - Comparison with other toolkits

2. Summary
Maemomm Overview

GTK+ C++ bindings

- **libsigc++** – signal library used throughout *mm projects
- **glibmm** – Glib bindings
- **cairomm** – Cairo bindings
- **pangomm** – Pango bindings
- **gtkmm** – GTK+ bindings
Maemo C++ bindings

- hildonmm – libhildon widgets
- hildon-fmmm – libhildonfm widgets
- libossomm – libosso bindings
- hildon-notifymm – libnotify bindings
- maemomm-documentation – Tutorial documentation and examples
Advantages of Maemomm

- Advantages of C++
- Less code
- Compile-time type-safety
- Inheritance used to derive new widgets
- Simple reference-counting
- Easier memory management
- UTF-8 string class
Advantages of C++

- Stricter type-checking
- Object-orientation as a language feature
- Construction and destruction
- The Standard Library
  - The Standard Template Library
- ...
Less code

- Language features make writing object-oriented code easier

More verbose code for C APIs

```c
hildon_button_set_title(
    HILDON_BUTTON(button),
    "Click Me");
```

Simpler C++ code, with less need for typecasts

```cpp
button.set_title("Click Me");
```
Greater type-safety

- libsigc++ provides compile-time type-safe callbacks
  - Cross-inspired by boost::signal
  - Less overhead compared with runtime signals
  - Argument binding and hiding
  - No additional preprocessor required

- Wrapped types do not need GObject-style typecast macros

- Namespaced enums and use of const
Inheritance to derive new widgets

Deriving a new window

class Demo : public Hildon::Window
{
    public:
        Demo();
        virtual ~Demo();

    private:
        // Other members...
};
Simple reference-counting

- **GObjects** are reference-counted types
  - Floating references
  - Container ownership
  - Manual referencing and unreferencing
- *With Maemomm*, just use the Glib::RefPtr<> smartpointer
Simple reference-counting

RefPtr example

```cpp
std::auto_ptr<Glib::Error> error;
Glib::RefPtr<Gtk::Builder> builder =
    Gtk::Builder::create_from_file("builder.ui", error);
```
Easier memory management

- Flexible memory management
  - Widgets as class members
  - Local scope widgets
  - Container ownership of widgets
  - Dynamic allocation of widgets
- No need to manually reference or unreference
Easier memory management

Memory management example

```cpp
// Function scope.
Gtk::Button button;

// Container ownership.
container.add(*Gtk::manage(
    new Gtk::Button("Click Me")));
```
UTF-8 string class

- **Glib::ustring** provides a UTF-8 wrapper around strings, with the `std::string` interface.
- Hildon and GTK+ use UTF-8 strings internally.
  - Use **Glib::ustring** for easy character-based iterator access to strings.
  - Cast to **std::string** if internationalization is not a concern.
  - Filenames always in native encoding, so return **std::string**.
Limitations compared to non-Maemo Gtkmm

- No exceptions
- No deprecated API
- No type-safe property accessors
- No default signal handlers
- No Atkmm API
## Comparisons with Qt

<table>
<thead>
<tr>
<th>Feature</th>
<th>Maemomm</th>
<th>Qt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Language</td>
<td>C++</td>
<td>C++ with moc macros</td>
</tr>
<tr>
<td>Containers</td>
<td>STL-compatible</td>
<td>Qt (STL-compatible)</td>
</tr>
<tr>
<td>Namespaces</td>
<td>Yes</td>
<td>Partial</td>
</tr>
<tr>
<td>Widget arrangement</td>
<td>Containers</td>
<td>Containers and layouts</td>
</tr>
<tr>
<td>Signal type-safety</td>
<td>Compile-time</td>
<td>Runtime</td>
</tr>
<tr>
<td>Hildon widgets</td>
<td>Yes</td>
<td>Partial</td>
</tr>
</tbody>
</table>
C++ bindings make life easier for C++ programmers

https://garage.maemo.org/projects/maemomm/

Future

- Hildon widgets in Maemo Harmattan?
- C++ bindings for GTK+ and Hildon in Maemo Harmattan?
Code examples go here
Widget code example screenshot