The Design of the WxOCaml library

An experiment with binding C++ libraries

Fabrice Le Fessant (INRIA/OCamlPro)
Workshop OCaml'2013
A new GUI Toolkit

- How to write GUIs in OCaml?
  - LablTK?
    - Pros: included in the distrib
    - Cons: bad look and feel, few widgets
  - LablGTK?
    - Pros: well tested, interface builder
    - Cons: no win64, not native on Win & MacOS
  - HTML5?
    - Pros: js_of_ocaml good, lots of JS libraries
    - Cons: webapp + http server, debug hard
- No Interface? Curses? Other ones?
Idea: binding for WxWidgets

- Good multi-platform support:
  - GTK under Linux, native on Windows and Mac OS
  - But the dev has been very slow in the last years :-(

- With bindings for MANY languages...
  - Very famous Python bindings
  - Also wxHaskell, wxEiffel, etc.
  - Except OCaml...
    - Not completely true : wxCaml, not finished
1st try: finishing WxCaml

- Going from OCaml to C++:
  - Use camlidl to generate stubs between C and OCaml from a "wxc.idl" file
  - C++ ↔ C stubs manually written ("elj" library)

- Problems:
  - Mostly untyped:
    - C stub arguments are not correctly typed
    - All widget types are equivalent!
  - WxCaml forked "wxc.idl" and "elj" to solve these problems, but they come from wxHaskell... that forked them from wxEiffel... unmaintainable!
2nd try: Goals

- Easy to maintain/extend:
  - No dependency towards wxHaskell or wxEiffel

- Easily accessible by beginners:
  - No fancy types: no Classes/Objects, no Polymorphic Variants, no GADT, no labels/optional arguments (for now...)
  - Should be usable from the first OCaml lesson !
  - Error messages for those are too complex to read
  - OO makes code unreadable with meth overloading

- Build a more abstract layer afterwards !
  - But write a few applications first...
Hello World
open WxClasses
open WxWidgets
open WxDefs

let _ =
  let OnInit (app : wxApp) =
    let frame = WxFrame.createAll None wxID_ANY
      "Hello World" (50, 50) (450, 350) wxDEFAULT_FRAME_STYLE in
    WxFrame.setIcon frame (WxIcon.createFromXPM Sample_xpm.sample_xpm);

  let about_id = wxID() in
  MENU_BAR.(wxFrame frame [
    "&File",
    Append(about_id, "&About");
    AppendSeparator();
    Append2(wxID_EXIT, "E&xit", "Exit from the application");
  ]);)
  ignore_wxStatusBar (WxFrame.createStatusBar frame);
  WxFrame.setStatusText frame "Welcome to wxWidgets!" 0;

  WxEVENT_TABLE.(wxFrame frame frame [
    EVT_MENU(wxID_EXIT, (fun _ _ -> exit 0));
    EVT_MENU(about_id, (fun _ _ ->
      ignore_int (WxMisc.wxMessageBoxAll
        "wxWidgets Hello World example."
        "About Hello World"
        (wxOK lor wxICON_INFORMATION)
        (Some (WxFrame.wxWindow frame))
        wxDefaultCoord wxDefaultCoord
      )))]);

  ignore_bool (WxFrame.show frame);
  WxApp.setTopWindow (WxFrame.wxWindow frame)

in
wxMain OnInit
Describes the C++ hierarchy of classes and their methods

```cpp
class wxTimer inherit wxEvtHandler begin

new Create (wxEvtHandler *owner, int id =-1 )
wxEvtHandler *GetOwner () const
void SetOwner (wxEvtHandler *owner, int id=-1)
bool Start (int milliseconds=-1, bool oneShot=false)

version 2.9 begin

  bool IsOneShot () const
  void Notify ()

end

end
```
For Each C++ class

- Two OCaml types:
  - `type wxTimer_class : the C++ object`
  - `type wxTimer = wxTimer_class wx : OCaml value`

- A module "WxTimer" with:
  - ALL its methods (including ancestors methods !)
    - "o->meth(x,y,z)" becomes
      "WxTimer.meth o x y z"
  - Safe coercions (identity) to all ancestors
  - An "Unsafe" sub-module, with coercions to all descendants (with runtime test)
For module WxTimer:

```
external create : wxEvtHandler -> int -> wxTimer = "wxTimer_Create_c"

[* methods of this class *]

external getOwner : wxTimer -> wxEvtHandler = "wxTimer_GetOwner_c"

external setOwner : wxTimer -> wxEvtHandler -> int -> unit = "wxTimer_SetOwner_c"

[* Methods inherited from parents, if any *]

external processEvent : wxTimer -> wxEvent -> bool = "wxEvtHandler_ProcessEvent_c"

[* Cast functions to parents *]

external wxEvtHandler : wxTimer -> wxEvtHandler = "%identity"

external wxObject : wxTimer -> wxObject = "%identity"
```
Dealing with C++ Objects

- C++ Objects are embedded in OCaml values as pairs (Class_ID, pointer)

```ocaml
value wxTimer_GetOwner_c(value self_v)
{
    CAMLparam0();
    CAMLLocal1(ret_v);
    wxTimer* self_c = (wxTimer*)Abstract_val(WXCLASS_wxTimer, self_v);
    wxEvtHandler * ret_c = self_c->GetOwner();
    ret_v = Val_abstract(WXCLASS_wxEvtHandler, (wxEvtHandler*) ret_c );
    CAMLreturn(ret_v);
}
```
Dealing with C++ Objects

- C++ Objects are embedded in OCaml values as pairs (Class_ID, pointer)
- For every method, only the ancestor stub is generated, with a generic cast

```ocaml
value wxEvtHandler_ProcessEvent_c(value self_v, value event_v_v) {
  CAMLparam0();
  CAMLlocal1(ret_v);
  wxEvtHandler* self_c =
      (wxEvtHandler*)Abstract_val(WXCLASS_wxEvtHandler,  self_v);
  wxEvent* event_c =
      (wxEvent*)Abstract_val(WXCLASS_wxEvent,  event_v);
  bool ret_c = self_c->ProcessEvent(*event_c);
  ret_v = Val_bool( ret_c);
  CAMLreturn(ret_v);
}
```
Dealing with C++ Objects

- A generic cast function is generated to perform C++ cast:

```c
extern "C" {
void* wxOCaml_cast(int dest_id, int src_id, void* ptr)
{
  if( dest_id == src_id) return ptr;
  if( ptr == NULL) return ptr;
  switch(dest_id * 167 + src_id){
  case 16375 : return (wxObject*)(wxAcceleratorTable*)ptr;
  case 8569 : return (wxEvent*)(wxActivateEvent*)ptr;
  case 16311 : return (wxEvtHandler*)(wxTimer*)ptr;
  case 16418 : return (wxObject*)(wxActivateEvent*)ptr;
  […]
  }
}
```
Dealing with Virtual Methods

- C++ classes can need method overriding:

```c++
class wxWizardPage inherit wxPanel begin
    wxBitmap GetBitmap() const
    wxWizardPage? GetPrev() const
    wxWizardPage? GetNext() const

    new Create (wxWizard? parent, const wxBitmap& bitmap)

    virtuals [
        (* These ones MUST be instantiated ! *)
        GetPrev, GetNext,
        (* These ones CAN be instantiated *)
        GetBitmap?,
        Validate? (* from wxWindow *)
    ]
end
```
OCaml constructors takes 2 extra arg: a record of methods and an initial state

Virtual methods take the state and this

```
[...]
let methods = WxVirtuals.WxOCamlWizardPage.({
  getPrev = (fun state this ->
    Some (WxOCamlWizardPage.wxWizardPage this));
  getNext = (fun state this -> None);
  getBitmap = Some (fun state this -> wxNullBitmap);
  validate = None;
}) in
let m_page1 = WxOCamlWizardPage.create methods initial_state (Some wizard) wxNullBitmap in
[...]
```
The “DSL + stub generator” approach works well for C++ libraries.

QT better than WxWidgets?
- The same approach would probably work!

Easy to extend WxOCaml:
- Currently, 90+ classes, 1600 C++ methods
- Write your WxOCaml application, and
- Add the classes/methods you need in the DSL

Web Site with GitHub link for sources: http://www.typerex.org/ocplib-wxOCaml/
Questions?