Future of Python Bindings

James Henstridge
james@daa.com.au

Python bindings Features

- Provide an object oriented python interface to GTK+ and related libraries.
- Takes care of reference counting and typecasting
- Complete enough to write full featured applications
- Used in several non trivial apps.

Architecture of Current Bindings

- C extension module wraps most functions in GTK+, making them callable from python code
  - majority of C code autogenerated from defs format
- Python code wraps procedural interface in a set of python classes that are used by the programmer

Problems with Current Bindings

- Possible to have multiple wrappers for each GtkObject
  - this model was chosen to avoid circular references
- object oriented interface built on top of procedural interface
- Non trivial ammount of python code to interpret on startup (takes time)
- Not easy to extend for extra libraries of widgets
- Procedural intermediate interface is not used much -- just takes up memory and diskspace

The Rewrite

- Chose to rewrite bindings
  - use ExtensionClass
  - new defs file format
  - new code generator
  - remove the procedural interface
  - add evil hack to give single wrapper per object
- Was not possible to keep compatibility with all the changes.
  - decide to target binding at GTK 2.0
  - abandoned ExtensionClass based GTK 1.2 binding available on `extension-class-branch` branch

ExtensionClass

- Types defined in an extension module are not classes (they can’t be subclassed)
- ExtensionClass is a module that allows creation of extension types that behave like classes
  - Written by Digital Creations, and used in Zope
- Allowed OOP interface to be implemented in extension module -- less python code.
- Version used with pygtk has some modifications
  - changes not merged back into official source (not for lack of trying).

New Defs Format

- Defs files are used by language bindings as prototypes for functions provided by the library
  - composed of s-expressions
  - used by Guile, Python, Perl and PHP gtk+ bindings
- Old defs format lacks some useful information
  - only defines object heirachy, and functions
  - doesn’t link functions to classes as methods
- New defs format has more information
  - original specification by Havoc
  - makes class <-> method relationships explicit
  - more type information
  - Takes guess work out of code generation

Code Generator

- New code generation tools needed
  - Take new style defs format as input
  - Output C code that uses ExtensionClass
- Add .h to defs conversion script
  - not perfect !
- Add extension module skeleton generator
- Easier to write bindings for new widget libraries
  - maybe 1/2 to 1 hour for a moderate size library
- Code generation tools will be installed with PyGTK

The Evil Hack

- makes sure that only a single wrapper will be used for a particular object for the lifetime of that object
- If some python code holds a reference to the wrapper, then the underlying object will not be freed
- Evil hack in destructor for wrapper that sometimes resurrects wrapper.
  - gets round circular reference problem
- Possible can make hack less evil using Python 2.1 weak references

GTK 2.0

- Base object system moved to glib
  - base type is now GObject
- more flexible signal emission system
- UTF-8 as internal encoding for user visible strings
- New GDK targets (framebuffer, win32)
- New widgets

GObject and PyGTK

- base object wrapper code separated out into the `gobject` module
- can be used independently of the `gtk` module
- Will support creation of new C level GObject types in the future.
- Currently does not interract with python threading model

GSignal and GClosures

- New signal system uses closure objects which wrap up a function, user data and marshallers.
- `gobject` module allows connecting handlers to signals, and adding signals to GObject classes.
- Signal system lets us set a closure as the class handler for a signal type
  - signals defined in python can do just as much as ones defined in C code.

Unicode

- Python 1.6 and 2.0 introduce unicode string type
- if a function expecting a normal string gets passed a unicode string, it gets converted to the "default" character encoding
- PyGTK sets default character encoding to UTF-8 at startup
  - you can pass unicode strings to PyGTK functions/methods, and they will be handled by GTK correctly
- Following code will work as expected:

```
w = gtk.GtkLabel(u’2\u03D6r’)
```

New GDK Targets

- GTK 1.2 has only an X11 backend
- GTK 2.0 introduces multiple backends
- PyGTK can be compiled with any of these backends
  - tested with x11 and linux-fb
  - should work with win32 as well
- Considering adding support for runtime selection of backend

Problems

- Requires unstable version of automake to build from CVS
  - should go away when automake 1.5 is released
- Requires patching of python interpreter or relinking of pixbuf loaders, pango modules and input method modules.
  - libtool 1.4 (CVS version) required for relinking
- Does not work with python threading at the moment
  - need a python threading guru to help fix this
  - problems with Python global interpreter lock

Todo

- complete bindings for all GTK APIs
- improve code generator
- documentation
- Gnome 2.0 bindings
  - wait til bindings are more complete